

GUIslice

0.17.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>GUIslice library</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>5</b>
3.1	Modules . . . . .	5
<b>4</b>	<b>Hierarchical Index</b>	<b>7</b>
4.1	Class Hierarchy . . . . .	7
<b>5</b>	<b>Data Structure Index</b>	<b>9</b>
5.1	Data Structures . . . . .	9
<b>6</b>	<b>File Index</b>	<b>11</b>
6.1	File List . . . . .	11
<b>7</b>	<b>Module Documentation</b>	<b>13</b>
7.1	General Functions . . . . .	13
7.1.1	Detailed Description . . . . .	14
7.1.2	Function Documentation . . . . .	14
7.1.2.1	gslc_DebugPrintf() . . . . .	14
7.1.2.2	gslc_GetClipRect() . . . . .	14
7.1.2.3	gslc_GetDriverDisp() . . . . .	15
7.1.2.4	gslc_GetDriverTouch() . . . . .	15
7.1.2.5	gslc_GetNameDisp() . . . . .	15
7.1.2.6	gslc_GetNameTouch() . . . . .	16

7.1.2.7	<a href="#">gslc_GetVer()</a>	16
7.1.2.8	<a href="#">gslc_GuiRotate()</a>	16
7.1.2.9	<a href="#">gslc_Init()</a>	17
7.1.2.10	<a href="#">gslc_InitDebug()</a>	18
7.1.2.11	<a href="#">gslc_Quit()</a>	18
7.1.2.12	<a href="#">gslc_SetBkgndColor()</a>	18
7.1.2.13	<a href="#">gslc_SetBkgndImage()</a>	19
7.1.2.14	<a href="#">gslc_SetClipRect()</a>	19
7.1.2.15	<a href="#">gslc_SetTransparentColor()</a>	20
7.1.2.16	<a href="#">gslc_Update()</a>	20
7.2	<a href="#">Graphics General Functions</a>	21
7.2.1	<a href="#">Detailed Description</a>	22
7.2.2	<a href="#">Function Documentation</a>	22
7.2.2.1	<a href="#">gslc_ClipLine()</a>	22
7.2.2.2	<a href="#">gslc_ClipPt()</a>	22
7.2.2.3	<a href="#">gslc_ClipRect()</a>	23
7.2.2.4	<a href="#">gslc_ColorBlend2()</a>	23
7.2.2.5	<a href="#">gslc_ColorBlend3()</a>	24
7.2.2.6	<a href="#">gslc_ColorEqual()</a>	24
7.2.2.7	<a href="#">gslc_cosFX()</a>	25
7.2.2.8	<a href="#">gslc_ExpandRect()</a>	25
7.2.2.9	<a href="#">gslc_GetImageFromFile()</a>	25
7.2.2.10	<a href="#">gslc_GetImageFromProg()</a>	26
7.2.2.11	<a href="#">gslc_GetImageFromRam()</a>	26
7.2.2.12	<a href="#">gslc_GetImageFromSD()</a>	27
7.2.2.13	<a href="#">gslc_InvalidateRgnAdd()</a>	27
7.2.2.14	<a href="#">gslc_InvalidateRgnPage()</a>	27
7.2.2.15	<a href="#">gslc_InvalidateRgnReset()</a>	28
7.2.2.16	<a href="#">gslc_InvalidateRgnScreen()</a>	28
7.2.2.17	<a href="#">gslc_IsInRect()</a>	28

7.2.2.18	<a href="#">gslc_IsInWH()</a>	29
7.2.2.19	<a href="#">gslc_PolarToXY()</a>	29
7.2.2.20	<a href="#">gslc_sinFX()</a>	30
7.2.2.21	<a href="#">gslc_UnionRect()</a>	30
7.3	<a href="#">Graphics Primitive Functions</a>	32
7.3.1	<a href="#">Detailed Description</a>	33
7.3.2	<a href="#">Function Documentation</a>	33
7.3.2.1	<a href="#">gslc_DrawFillCircle()</a>	33
7.3.2.2	<a href="#">gslc_DrawFillGradSector()</a>	33
7.3.2.3	<a href="#">gslc_DrawFillQuad()</a>	34
7.3.2.4	<a href="#">gslc_DrawFillRect()</a>	35
7.3.2.5	<a href="#">gslc_DrawFillRoundRect()</a>	35
7.3.2.6	<a href="#">gslc_DrawFillSector()</a>	36
7.3.2.7	<a href="#">gslc_DrawFillTriangle()</a>	36
7.3.2.8	<a href="#">gslc_DrawFrameCircle()</a>	37
7.3.2.9	<a href="#">gslc_DrawFrameQuad()</a>	37
7.3.2.10	<a href="#">gslc_DrawFrameRect()</a>	38
7.3.2.11	<a href="#">gslc_DrawFrameRoundRect()</a>	38
7.3.2.12	<a href="#">gslc_DrawFrameTriangle()</a>	39
7.3.2.13	<a href="#">gslc_DrawLine()</a>	39
7.3.2.14	<a href="#">gslc_DrawLineH()</a>	40
7.3.2.15	<a href="#">gslc_DrawLinePolar()</a>	40
7.3.2.16	<a href="#">gslc_DrawLineV()</a>	41
7.3.2.17	<a href="#">gslc_DrawSetPixel()</a>	42
7.4	<a href="#">Font Functions</a>	43
7.4.1	<a href="#">Detailed Description</a>	43
7.4.2	<a href="#">Function Documentation</a>	43
7.4.2.1	<a href="#">gslc_FontAdd()</a>	43
7.4.2.2	<a href="#">gslc_FontGet()</a>	44
7.4.2.3	<a href="#">gslc_FontSet()</a>	44

7.4.2.4	<code>gslc_FontSetMode()</code>	45
7.5	Page Functions	46
7.5.1	Detailed Description	46
7.5.2	Function Documentation	46
7.5.2.1	<code>gslc_GetPageCur()</code>	46
7.5.2.2	<code>gslc_PageAdd()</code>	47
7.5.2.3	<code>gslc_PageFindElemById()</code>	47
7.5.2.4	<code>gslc_PageRedrawGet()</code>	48
7.5.2.5	<code>gslc_PageRedrawSet()</code>	48
7.5.2.6	<code>gslc_PopupHide()</code>	49
7.5.2.7	<code>gslc_PopupShow()</code>	49
7.5.2.8	<code>gslc_SetPageBase()</code>	50
7.5.2.9	<code>gslc_SetPageCur()</code>	50
7.5.2.10	<code>gslc_SetPageOverlay()</code>	50
7.5.2.11	<code>gslc_SetStackPage()</code>	51
7.5.2.12	<code>gslc_SetStackState()</code>	51
7.6	Element Functions	53
7.6.1	Detailed Description	53
7.7	Element: Creation Functions	54
7.7.1	Detailed Description	54
7.7.2	Function Documentation	54
7.7.2.1	<code>gslc_ElemCreateBox()</code>	55
7.7.2.2	<code>gslc_ElemCreateBtnImg()</code>	56
7.7.2.3	<code>gslc_ElemCreateBtnTxt()</code>	57
7.7.2.4	<code>gslc_ElemCreateImg()</code>	57
7.7.2.5	<code>gslc_ElemCreateLine()</code>	58
7.7.2.6	<code>gslc_ElemCreateTxt()</code>	59
7.8	Element: General Functions	60
7.8.1	Detailed Description	60
7.8.2	Function Documentation	60

7.8.2.1	<a href="#">gslc_ElemGetId()</a>	60
7.9	<a href="#">Element: Update Functions</a>	61
7.9.1	<a href="#">Detailed Description</a>	63
7.9.2	<a href="#">Function Documentation</a>	63
7.9.2.1	<a href="#">gslc_ElemCalcRectState()</a>	63
7.9.2.2	<a href="#">gslc_ElemCalcResizeForFocus()</a>	64
7.9.2.3	<a href="#">gslc_ElemGetEdit()</a>	64
7.9.2.4	<a href="#">gslc_ElemGetFocus()</a>	64
7.9.2.5	<a href="#">gslc_ElemGetFocusEn()</a>	65
7.9.2.6	<a href="#">gslc_ElemGetGlow()</a>	65
7.9.2.7	<a href="#">gslc_ElemGetGlowEn()</a>	66
7.9.2.8	<a href="#">gslc_ElemGetGroup()</a>	66
7.9.2.9	<a href="#">gslc_ElemGetOnScreen()</a>	66
7.9.2.10	<a href="#">gslc_ElemGetRect()</a>	67
7.9.2.11	<a href="#">gslc_ElemGetRedraw()</a>	67
7.9.2.12	<a href="#">gslc_ElemGetTxtStr()</a>	68
7.9.2.13	<a href="#">gslc_ElemGetVisible()</a>	68
7.9.2.14	<a href="#">gslc_ElemGrowRect()</a>	69
7.9.2.15	<a href="#">gslc_ElemOwnsCoord()</a>	69
7.9.2.16	<a href="#">gslc_ElemSetClickEn()</a>	70
7.9.2.17	<a href="#">gslc_ElemSetCol()</a>	70
7.9.2.18	<a href="#">gslc_ElemSetDrawFunc()</a>	70
7.9.2.19	<a href="#">gslc_ElemSetEdit()</a>	71
7.9.2.20	<a href="#">gslc_ElemSetFillEn()</a>	71
7.9.2.21	<a href="#">gslc_ElemSetFocus()</a>	72
7.9.2.22	<a href="#">gslc_ElemSetFocusEn()</a>	72
7.9.2.23	<a href="#">gslc_ElemSetFrameEn()</a>	73
7.9.2.24	<a href="#">gslc_ElemSetGlow()</a>	73
7.9.2.25	<a href="#">gslc_ElemSetGlowCol()</a>	74
7.9.2.26	<a href="#">gslc_ElemSetGlowEn()</a>	74

7.9.2.27	<a href="#">gslc_ElemSetGroup()</a>	75
7.9.2.28	<a href="#">gslc_ElemSetRect()</a>	75
7.9.2.29	<a href="#">gslc_ElemSetRedraw()</a>	76
7.9.2.30	<a href="#">gslc_ElemSetRoundEn()</a>	76
7.9.2.31	<a href="#">gslc_ElemSetStyleFrom()</a>	76
7.9.2.32	<a href="#">gslc_ElemSetTickFunc()</a>	77
7.9.2.33	<a href="#">gslc_ElemSetTouchFunc()</a>	77
7.9.2.34	<a href="#">gslc_ElemSetTxtAlign()</a>	78
7.9.2.35	<a href="#">gslc_ElemSetTxtCol()</a>	78
7.9.2.36	<a href="#">gslc_ElemSetTxtEnc()</a>	79
7.9.2.37	<a href="#">gslc_ElemSetTxtMargin()</a>	79
7.9.2.38	<a href="#">gslc_ElemSetTxtMarginXY()</a>	80
7.9.2.39	<a href="#">gslc_ElemSetTxtMem()</a>	80
7.9.2.40	<a href="#">gslc_ElemSetTxtStr()</a>	81
7.9.2.41	<a href="#">gslc_ElemSetVisible()</a>	81
7.9.2.42	<a href="#">gslc_ElemUpdateFont()</a>	81
7.9.2.43	<a href="#">gslc_ResetRectState()</a>	82
7.9.2.44	<a href="#">gslc_StrCopy()</a>	82
7.10	<a href="#">Touchscreen Functions</a>	84
7.10.1	<a href="#">Detailed Description</a>	84
7.10.2	<a href="#">Macro Definition Documentation</a>	84
7.10.2.1	<a href="#">TOUCH_ROTATION_DATA [1/2]</a>	85
7.10.2.2	<a href="#">TOUCH_ROTATION_DATA [2/2]</a>	85
7.10.2.3	<a href="#">TOUCH_ROTATION_FLIPX [1/2]</a>	85
7.10.2.4	<a href="#">TOUCH_ROTATION_FLIPX [2/2]</a>	85
7.10.2.5	<a href="#">TOUCH_ROTATION_FLIPY [1/2]</a>	85
7.10.2.6	<a href="#">TOUCH_ROTATION_FLIPY [2/2]</a>	85
7.10.2.7	<a href="#">TOUCH_ROTATION_SWAPXY [1/2]</a>	86
7.10.2.8	<a href="#">TOUCH_ROTATION_SWAPXY [2/2]</a>	86
7.10.3	<a href="#">Function Documentation</a>	86



7.10.3.1	<a href="#">gslc_GetTouch()</a>	86
7.10.3.2	<a href="#">gslc_GetTouchEn()</a>	86
7.10.3.3	<a href="#">gslc_InitTouch()</a>	87
7.10.3.4	<a href="#">gslc_SetTouchEn()</a>	87
7.10.3.5	<a href="#">gslc_SetTouchPressCal()</a>	88
7.10.3.6	<a href="#">gslc_SetTouchRemapCal()</a>	88
7.10.3.7	<a href="#">gslc_SetTouchRemapEn()</a>	89
7.10.3.8	<a href="#">gslc_SetTouchRemapYX()</a>	89
7.11	<a href="#">Input Mapping Functions</a>	90
7.11.1	<a href="#">Detailed Description</a>	90
7.11.2	<a href="#">Function Documentation</a>	90
7.11.2.1	<a href="#">gslc_FocusElemGet()</a>	90
7.11.2.2	<a href="#">gslc_FocusElemIndSet()</a>	91
7.11.2.3	<a href="#">gslc_FocusElemStep()</a>	91
7.11.2.4	<a href="#">gslc_FocusPageStep()</a>	92
7.11.2.5	<a href="#">gslc_FocusSetToTrackedElem()</a>	92
7.11.2.6	<a href="#">gslc_InitInputMap()</a>	92
7.11.2.7	<a href="#">gslc_InputMapAdd()</a>	93
7.11.2.8	<a href="#">gslc_SetPinPollFunc()</a>	93
7.12	<a href="#">General Purpose Macros</a>	95
7.12.1	<a href="#">Detailed Description</a>	95
7.12.2	<a href="#">Macro Definition Documentation</a>	95
7.12.2.1	<a href="#">GSLC_DEBUG2_PRINT</a>	95
7.12.2.2	<a href="#">GSLC_DEBUG2_PRINT_CONST</a>	95
7.12.2.3	<a href="#">GSLC_DEBUG_PRINT</a>	95
7.12.2.4	<a href="#">GSLC_DEBUG_PRINT_CONST</a>	96
7.13	<a href="#">Flash-based Element Macros</a>	97
7.13.1	<a href="#">Detailed Description</a>	97
7.13.2	<a href="#">Macro Definition Documentation</a>	97
7.13.2.1	<a href="#">gslc_ElemCreateBox_P</a>	97

7.13.2.2	gslc_ElemCreateBtnTxt_P . . . . .	98
7.13.2.3	gslc_ElemCreateLine_P . . . . .	99
7.13.2.4	gslc_ElemCreateTxt_P . . . . .	99
7.13.2.5	gslc_ElemCreateTxt_P_R . . . . .	100
7.13.2.6	gslc_ElemCreateTxt_P_R_ext . . . . .	101
7.14	Internal Functions . . . . .	103
7.14.1	Detailed Description . . . . .	109
7.14.2	Variable Documentation . . . . .	109
7.14.2.1	abPageStackActive . . . . .	109
7.14.2.2	abPageStackDoDraw . . . . .	109
7.14.2.3	apPageStack . . . . .	110
7.14.2.4	asElem . . . . .	110
7.14.2.5	asElemRef . . . . .	110
7.14.2.6	asFont . . . . .	110
7.14.2.7	asInputMap . . . . .	110
7.14.2.8	asPage . . . . .	110
7.14.2.9	b . . . . .	111
7.14.2.10	bEventPending . . . . .	111
7.14.2.11	blnvalidateEn . . . . .	111
7.14.2.12	bRedrawNeeded . . . . .	111
7.14.2.13	bRedrawPartialEn . . . . .	111
7.14.2.14	bScreenNeedFlip . . . . .	111
7.14.2.15	bScreenNeedRedraw . . . . .	112
7.14.2.16	bTouchEn . . . . .	112
7.14.2.17	bTouchRemapEn . . . . .	112
7.14.2.18	bTouchRemapYX . . . . .	112
7.14.2.19	colBack . . . . .	112
7.14.2.20	colElemFill . . . . .	112
7.14.2.21	colElemFillGlow . . . . .	113
7.14.2.22	colElemFrame . . . . .	113

7.14.2.23 colElemFrameGlow . . . . .	113
7.14.2.24 colElemText . . . . .	113
7.14.2.25 colElemTextGlow . . . . .	113
7.14.2.26 colFocus [1/2] . . . . .	113
7.14.2.27 colFocus [2/2] . . . . .	114
7.14.2.28 colFocusEdit . . . . .	114
7.14.2.29 colFocusNone . . . . .	114
7.14.2.30 colFrm . . . . .	114
7.14.2.31 collInner . . . . .	114
7.14.2.32 colTxtBack . . . . .	114
7.14.2.33 colTxtFore . . . . .	114
7.14.2.34 eAction . . . . .	115
7.14.2.35 eElemFlags . . . . .	115
7.14.2.36 eEvent . . . . .	115
7.14.2.37 eFontRefMode . . . . .	115
7.14.2.38 eFontRefType . . . . .	115
7.14.2.39 elmgFlags . . . . .	115
7.14.2.40 elnitStatTouch . . . . .	116
7.14.2.41 eTouch . . . . .	116
7.14.2.42 eTxtAlign . . . . .	116
7.14.2.43 eTxtFlags . . . . .	116
7.14.2.44 eType . . . . .	116
7.14.2.45 g . . . . .	116
7.14.2.46 h . . . . .	117
7.14.2.47 nActionVal . . . . .	117
7.14.2.48 nDisp0H . . . . .	117
7.14.2.49 nDisp0W . . . . .	117
7.14.2.50 nDispDepth . . . . .	117
7.14.2.51 nDispH . . . . .	117
7.14.2.52 nDispW . . . . .	118

7.14.2.53 nElemAutoldNext . . . . .	118
7.14.2.54 nElemCnt . . . . .	118
7.14.2.55 nElemIndTracked . . . . .	118
7.14.2.56 nElemMax . . . . .	118
7.14.2.57 nElemRefCnt . . . . .	118
7.14.2.58 nElemRefMax . . . . .	119
7.14.2.59 nFeatures . . . . .	119
7.14.2.60 nFlipX . . . . .	119
7.14.2.61 nFlipY . . . . .	119
7.14.2.62 nFocusElemInd . . . . .	119
7.14.2.63 nFocusElemMax . . . . .	119
7.14.2.64 nFocusPageInd . . . . .	120
7.14.2.65 nFocusSavedElemInd . . . . .	120
7.14.2.66 nFocusSavedPageInd . . . . .	120
7.14.2.67 nFontCnt . . . . .	120
7.14.2.68 nFontMax . . . . .	120
7.14.2.69 nFrameRateCnt . . . . .	120
7.14.2.70 nFrameRateStart . . . . .	121
7.14.2.71 nGroup . . . . .	121
7.14.2.72 nId [1/2] . . . . .	121
7.14.2.73 nId [2/2] . . . . .	121
7.14.2.74 nInputMapCnt . . . . .	121
7.14.2.75 nInputMapMax . . . . .	121
7.14.2.76 nInputMode . . . . .	122
7.14.2.77 nPageCnt . . . . .	122
7.14.2.78 nPageId . . . . .	122
7.14.2.79 nPageMax . . . . .	122
7.14.2.80 nRotation . . . . .	122
7.14.2.81 nRoundRadius . . . . .	122
7.14.2.82 nSize . . . . .	123

7.14.2.83 nStrBufMax	123
7.14.2.84 nSubType	123
7.14.2.85 nSwapXY	123
7.14.2.86 nTouchCalPressMax	123
7.14.2.87 nTouchCalPressMin	123
7.14.2.88 nTouchCalXMax	124
7.14.2.89 nTouchCalXMin	124
7.14.2.90 nTouchCalYMax	124
7.14.2.91 nTouchCalYMin	124
7.14.2.92 nTouchLastPress	124
7.14.2.93 nTouchLastX	124
7.14.2.94 nTouchLastY	125
7.14.2.95 nTouchRotation	125
7.14.2.96 nTxtMarginX	125
7.14.2.97 nTxtMarginY	125
7.14.2.98 nType	125
7.14.2.99 nVal	125
7.14.2.100nX	126
7.14.2.101nY	126
7.14.2.102pElem	126
7.14.2.103pElemRefParent	126
7.14.2.104pElemRefTracked	126
7.14.2.105pFname	126
7.14.2.106pFocusElemRef	127
7.14.2.107pFocusPage	127
7.14.2.108pfuncPinPoll	127
7.14.2.109pfuncXDraw	127
7.14.2.110pfuncXEvent	127
7.14.2.111pfuncXTick	127
7.14.2.112pfuncXTouch	128

7.14.2.113pImgBuf . . . . .	128
7.14.2.114pStrBuf . . . . .	128
7.14.2.115pTxtFont . . . . .	128
7.14.2.116pvData . . . . .	128
7.14.2.117pvDriver . . . . .	128
7.14.2.118pvFont . . . . .	129
7.14.2.119pvImgRaw . . . . .	129
7.14.2.120pvScope . . . . .	129
7.14.2.121pXData . . . . .	129
7.14.2.122 . . . . .	129
7.14.2.123Bounds . . . . .	129
7.14.2.124Elem . . . . .	130
7.14.2.125Focus . . . . .	130
7.14.2.126Full . . . . .	130
7.14.2.127Inner . . . . .	130
7.14.2.128InvalidateRect . . . . .	130
7.14.2.129sCollect . . . . .	130
7.14.2.130sElemRefTmp . . . . .	130
7.14.2.131sElemTmp . . . . .	131
7.14.2.132sElemTmpProg . . . . .	131
7.14.2.133sEventPend . . . . .	131
7.14.2.134sEventTouchPend . . . . .	131
7.14.2.135sImgRefBkgnd . . . . .	131
7.14.2.136sImgRefGlow . . . . .	131
7.14.2.137sImgRefNorm . . . . .	132
7.14.2.138sTransCol . . . . .	132
7.14.2.139w . . . . .	132
7.14.2.140x [1/2] . . . . .	132
7.14.2.141x [2/2] . . . . .	132
7.14.2.142y [1/2] . . . . .	132

7.14.2.143y [2/2] . . . . .	132
7.15 Internal: Misc Functions . . . . .	133
7.15.1 Detailed Description . . . . .	133
7.15.2 Function Documentation . . . . .	133
7.15.2.1 gslc_ResetImage() . . . . .	133
7.16 Internal: Element Functions . . . . .	134
7.16.1 Detailed Description . . . . .	135
7.16.2 Function Documentation . . . . .	135
7.16.2.1 gslc_DrawTxtBase() . . . . .	135
7.16.2.2 gslc_ElemAdd() . . . . .	135
7.16.2.3 gslc_ElemCreate() . . . . .	136
7.16.2.4 gslc_ElemDraw() . . . . .	137
7.16.2.5 gslc_ElemDrawByRef() . . . . .	137
7.16.2.6 gslc_ElemSetImage() . . . . .	138
7.16.2.7 gslc_GetElemFromRef() . . . . .	138
7.16.2.8 gslc_GetElemFromRefD() . . . . .	139
7.16.2.9 gslc_GetElemRefFlag() . . . . .	139
7.16.2.10 gslc_GetXDataFromRef() . . . . .	139
7.16.2.11 gslc_SetElemRefFlag() . . . . .	140
7.16.2.12 gslc_SetFocusCol() . . . . .	140
7.16.2.13 gslc_SetRoundRadius() . . . . .	141
7.17 Internal: Page Functions . . . . .	142
7.17.1 Detailed Description . . . . .	142
7.17.2 Function Documentation . . . . .	142
7.17.2.1 gslc_ElemEvent() . . . . .	142
7.17.2.2 gslc_ElemSendEventTouch() . . . . .	143
7.17.2.3 gslc_EventCreate() . . . . .	143
7.17.2.4 gslc_PageEvent() . . . . .	144
7.17.2.5 gslc_PageFindById() . . . . .	144
7.17.2.6 gslc_PageFlipGet() . . . . .	145

7.17.2.7	<a href="#">gslc_PageFlipGo()</a>	145
7.17.2.8	<a href="#">gslc_PageFlipSet()</a>	145
7.17.2.9	<a href="#">gslc_PageRedrawCalc()</a>	146
7.17.2.10	<a href="#">gslc_PageRedrawGo()</a>	146
7.18	Internal: Element Collection Functions	148
7.18.1	Detailed Description	148
7.18.2	Function Documentation	148
7.18.2.1	<a href="#">gslc_CollectElemAdd()</a>	149
7.18.2.2	<a href="#">gslc_CollectFindElemById()</a>	149
7.18.2.3	<a href="#">gslc_CollectFindElemFromCoord()</a>	150
7.18.2.4	<a href="#">gslc_CollectGetElemRefTracked()</a>	150
7.18.2.5	<a href="#">gslc_CollectGetNextId()</a>	151
7.18.2.6	<a href="#">gslc_CollectGetRedraw()</a>	151
7.18.2.7	<a href="#">gslc_CollectReset()</a>	151
7.18.2.8	<a href="#">gslc_CollectSetElemTracked()</a>	152
7.18.2.9	<a href="#">gslc_CollectSetParent()</a>	152
7.19	Internal: Element Collection Event Functions	154
7.19.1	Detailed Description	154
7.19.2	Function Documentation	154
7.19.2.1	<a href="#">gslc_CollectEvent()</a>	154
7.19.2.2	<a href="#">gslc_CollectInput()</a>	155
7.19.2.3	<a href="#">gslc_CollectTouch()</a>	155
7.19.2.4	<a href="#">gslc_CollectTouchCompound()</a>	156
7.20	Internal: Tracking Functions	157
7.20.1	Detailed Description	157
7.20.2	Function Documentation	157
7.20.2.1	<a href="#">gslc_InputMapLookup()</a>	157
7.20.2.2	<a href="#">gslc_TrackInput()</a>	158
7.20.2.3	<a href="#">gslc_TrackTouch()</a>	158
7.21	Internal: Cleanup Functions	160
7.21.1	Detailed Description	160
7.21.2	Function Documentation	160
7.21.2.1	<a href="#">gslc_CollectDestruct()</a>	160
7.21.2.2	<a href="#">gslc_ElemDestruct()</a>	161
7.21.2.3	<a href="#">gslc_GuiDestruct()</a>	161
7.21.2.4	<a href="#">gslc_PageDestruct()</a>	161
7.21.2.5	<a href="#">gslc_ResetElem()</a>	163
7.21.2.6	<a href="#">gslc_ResetFont()</a>	163



<b>8 Data Structure Documentation</b>	<b>165</b>
8.1 gslc_tsCollect Struct Reference	165
8.1.1 Detailed Description	166
8.2 gslc_tsColor Struct Reference	166
8.2.1 Detailed Description	166
8.3 gslc_tsDriver Struct Reference	166
8.3.1 Field Documentation	167
8.3.1.1 nColBkgnd	167
8.3.1.2 pvFontLast	167
8.3.1.3 rClipRect	167
8.4 gslc_tsElem Struct Reference	168
8.4.1 Detailed Description	169
8.5 gslc_tsElemRef Struct Reference	170
8.5.1 Detailed Description	170
8.6 gslc_tsEvent Struct Reference	170
8.6.1 Detailed Description	171
8.7 gslc_tsEventTouch Struct Reference	171
8.7.1 Detailed Description	171
8.8 gslc_tsFont Struct Reference	171
8.8.1 Detailed Description	172
8.9 gslc_tsGui Struct Reference	172
8.9.1 Detailed Description	175
8.10 gslc_tsImgRef Struct Reference	175
8.10.1 Detailed Description	175
8.11 gslc_tsInputMap Struct Reference	176
8.11.1 Detailed Description	176
8.12 gslc_tsKey Struct Reference	176
8.12.1 Detailed Description	177
8.12.2 Field Documentation	177
8.12.2.1 nCol	177

8.12.2.2	<a href="#">nColSpan</a>	177
8.12.2.3	<a href="#">nId</a>	177
8.12.2.4	<a href="#">nRow</a>	177
8.12.2.5	<a href="#">nRowSpan</a>	177
8.12.2.6	<a href="#">nType</a>	178
8.13	<a href="#">gslc_tsLabelSpecial Struct Reference</a>	178
8.13.1	<a href="#">Detailed Description</a>	178
8.13.2	<a href="#">Field Documentation</a>	178
8.13.2.1	<a href="#">nId</a>	178
8.13.2.2	<a href="#">pLabel</a>	178
8.14	<a href="#">gslc_tsPage Struct Reference</a>	179
8.14.1	<a href="#">Detailed Description</a>	179
8.15	<a href="#">gslc_tsPt Struct Reference</a>	180
8.15.1	<a href="#">Detailed Description</a>	180
8.16	<a href="#">gslc_tsRect Struct Reference</a>	180
8.16.1	<a href="#">Detailed Description</a>	180
8.17	<a href="#">gslc_tsRectState Struct Reference</a>	181
8.17.1	<a href="#">Detailed Description</a>	181
8.18	<a href="#">gslc_tsXCheckbox Struct Reference</a>	182
8.18.1	<a href="#">Detailed Description</a>	182
8.18.2	<a href="#">Field Documentation</a>	182
8.18.2.1	<a href="#">bChecked</a>	182
8.18.2.2	<a href="#">bRadio</a>	183
8.18.2.3	<a href="#">colCheck</a>	183
8.18.2.4	<a href="#">nStyle</a>	183
8.18.2.5	<a href="#">pfuncXToggle</a>	183
8.19	<a href="#">gslc_tsXGauge Struct Reference</a>	183
8.19.1	<a href="#">Detailed Description</a>	184
8.19.2	<a href="#">Field Documentation</a>	184
8.19.2.1	<a href="#">bFlip</a>	184

8.19.2.2	bIndicFill	185
8.19.2.3	bValLastValid	185
8.19.2.4	bVert	185
8.19.2.5	colGauge	185
8.19.2.6	colTick	185
8.19.2.7	nIndicLen	185
8.19.2.8	nIndicTip	186
8.19.2.9	nMax	186
8.19.2.10	nMin	186
8.19.2.11	nStyle	186
8.19.2.12	nTickCnt	186
8.19.2.13	nTickLen	186
8.19.2.14	nVal	187
8.19.2.15	nValLast	187
8.20	gslc_tsXGlowball Struct Reference	187
8.20.1	Detailed Description	188
8.20.2	Field Documentation	188
8.20.2.1	colBg	188
8.20.2.2	nAngEnd	188
8.20.2.3	nAngStart	189
8.20.2.4	nMidX	189
8.20.2.5	nMidY	189
8.20.2.6	nNumRings	189
8.20.2.7	nQuality	189
8.20.2.8	nVal	189
8.20.2.9	nValLast	190
8.20.2.10	pRings	190
8.21	gslc_tsXGlowballRing Struct Reference	190
8.21.1	Field Documentation	190
8.21.1.1	cCol	191

8.21.1.2	nRad1	191
8.21.1.3	nRad2	191
8.22	gslc_tsXGraph Struct Reference	191
8.22.1	Detailed Description	192
8.22.2	Field Documentation	192
8.22.2.1	bScrollEn	192
8.22.2.2	colGraph	193
8.22.2.3	eStyle	193
8.22.2.4	nBufCnt	193
8.22.2.5	nBufMax	193
8.22.2.6	nMargin	193
8.22.2.7	nPlotIndMax	193
8.22.2.8	nPlotIndStart	194
8.22.2.9	nPlotValMax	194
8.22.2.10	nPlotValMin	194
8.22.2.11	nScrollPos	194
8.22.2.12	nWndHeight	194
8.22.2.13	nWndWidth	194
8.22.2.14	pBuf	195
8.23	gslc_tsXKeyPad Struct Reference	195
8.23.1	Detailed Description	196
8.23.2	Field Documentation	196
8.23.2.1	acBuffer	196
8.23.2.2	nBufferLen	196
8.23.2.3	nBufferMax	196
8.23.2.4	nCursorPos	196
8.23.2.5	nFocusKeyInd	196
8.23.2.6	nGlowKeyInd	197
8.23.2.7	nScrollPos	197
8.23.2.8	pConfig	197

8.23.2.9	pfuncCb	197
8.23.2.10	pTargetRef	197
8.23.2.11	sRedraw	197
8.24	gslc_tsXKeyPadCfg Struct Reference	198
8.24.1	Detailed Description	199
8.24.2	Field Documentation	199
8.24.2.1	bRoundEn	199
8.24.2.2	eLayoutDef	199
8.24.2.3	eLayoutSel	199
8.24.2.4	nButtonSpaceX	200
8.24.2.5	nButtonSpaceY	200
8.24.2.6	nButtonSzH	200
8.24.2.7	nButtonSzW	200
8.24.2.8	nDispMax	200
8.24.2.9	nFontId	200
8.24.2.10	nFrameMargin	201
8.24.2.11	nMaxCols	201
8.24.2.12	nMaxRows	201
8.24.2.13	nOffsetX	201
8.24.2.14	nOffsetY	201
8.24.2.15	pfuncBtnEvt	201
8.24.2.16	pfuncLabelGet	202
8.24.2.17	pfuncReset	202
8.24.2.18	pfuncStyleGet	202
8.24.2.19	pfuncTxtInit	202
8.24.2.20	pLayout	202
8.24.2.21	pLayouts	202
8.25	gslc_tsXKeyPadCfg_Alpha Struct Reference	203
8.25.1	Field Documentation	203
8.25.1.1	sBaseCfg	203

8.26	<a href="#">gslc_tsXKeyPadCfg_Num Struct Reference</a>	204
8.26.1	<a href="#">Field Documentation</a>	204
8.26.1.1	<a href="#">bFloatEn</a>	204
8.26.1.2	<a href="#">bSignEn</a>	205
8.26.1.3	<a href="#">bValDecimalPt</a>	205
8.26.1.4	<a href="#">bValPositive</a>	205
8.26.1.5	<a href="#">sBaseCfg</a>	205
8.27	<a href="#">gslc_tsXKeyPadData Struct Reference</a>	205
8.27.1	<a href="#">Detailed Description</a>	206
8.27.2	<a href="#">Field Documentation</a>	206
8.27.2.1	<a href="#">pStr</a>	206
8.27.2.2	<a href="#">pTargetRef</a>	206
8.28	<a href="#">gslc_tsXKeyPadResult Struct Reference</a>	206
8.28.1	<a href="#">Detailed Description</a>	207
8.28.2	<a href="#">Field Documentation</a>	207
8.28.2.1	<a href="#">eRedrawState</a>	207
8.28.2.2	<a href="#">nRedrawKeyId1</a>	207
8.28.2.3	<a href="#">nRedrawKeyId2</a>	207
8.29	<a href="#">gslc_tsXListbox Struct Reference</a>	208
8.29.1	<a href="#">Detailed Description</a>	209
8.29.2	<a href="#">Field Documentation</a>	209
8.29.2.1	<a href="#">bFocusLast</a>	209
8.29.2.2	<a href="#">bGlowLast</a>	209
8.29.2.3	<a href="#">bItemAutoSizeH</a>	209
8.29.2.4	<a href="#">bItemAutoSizeW</a>	210
8.29.2.5	<a href="#">bNeedRecalc</a>	210
8.29.2.6	<a href="#">colGap</a>	210
8.29.2.7	<a href="#">nBufItemsMax</a>	210
8.29.2.8	<a href="#">nBufItemsPos</a>	210
8.29.2.9	<a href="#">nCols</a>	210

8.29.2.10 nItemCnt . . . . .	211
8.29.2.11 nItemCurSel . . . . .	211
8.29.2.12 nItemCurSelLast . . . . .	211
8.29.2.13 nItemGap . . . . .	211
8.29.2.14 nItemH . . . . .	211
8.29.2.15 nItemSavedSel . . . . .	211
8.29.2.16 nItemTop . . . . .	212
8.29.2.17 nItemW . . . . .	212
8.29.2.18 nMarginH . . . . .	212
8.29.2.19 nMarginW . . . . .	212
8.29.2.20 nRows . . . . .	212
8.29.2.21 pBufItems . . . . .	212
8.29.2.22 pfuncXSel . . . . .	213
8.30 gslc_tsXProgress Struct Reference . . . . .	213
8.30.1 Detailed Description . . . . .	214
8.30.2 Field Documentation . . . . .	214
8.30.2.1 bFlip . . . . .	214
8.30.2.2 bValLastValid . . . . .	214
8.30.2.3 bVert . . . . .	214
8.30.2.4 colGauge . . . . .	214
8.30.2.5 nMax . . . . .	214
8.30.2.6 nMin . . . . .	215
8.30.2.7 nVal . . . . .	215
8.30.2.8 nValLast . . . . .	215
8.31 gslc_tsXRadial Struct Reference . . . . .	215
8.31.1 Detailed Description . . . . .	216
8.31.2 Field Documentation . . . . .	216
8.31.2.1 bFlip . . . . .	216
8.31.2.2 bIndicFill . . . . .	217
8.31.2.3 bValLastValid . . . . .	217

8.31.2.4	colGauge	217
8.31.2.5	colTick	217
8.31.2.6	nIndicLen	217
8.31.2.7	nIndicTip	217
8.31.2.8	nMax	218
8.31.2.9	nMin	218
8.31.2.10	nTickCnt	218
8.31.2.11	nTickLen	218
8.31.2.12	nVal	218
8.31.2.13	nValLast	218
8.32	gslc_tsXRamp Struct Reference	219
8.32.1	Detailed Description	219
8.32.2	Field Documentation	219
8.32.2.1	bValLastValid	219
8.32.2.2	nMax	219
8.32.2.3	nMin	220
8.32.2.4	nVal	220
8.32.2.5	nValLast	220
8.33	gslc_tsXRingGauge Struct Reference	220
8.33.1	Detailed Description	221
8.33.2	Field Documentation	221
8.33.2.1	acStrLast	221
8.33.2.2	bGradient	221
8.33.2.3	colRing1	221
8.33.2.4	colRing2	222
8.33.2.5	colRingRemain	222
8.33.2.6	nAngRange	222
8.33.2.7	nAngStart	222
8.33.2.8	nQuality	222
8.33.2.9	nSegGap	222



8.33.2.10 nThickness	222
8.33.2.11 nVal	223
8.33.2.12 nValLast	223
8.33.2.13 nValMax	223
8.33.2.14 nValMin	223
8.34 gslc_tsXSeekBar Struct Reference	223
8.34.1 Detailed Description	224
8.34.2 Field Documentation	224
8.34.2.1 bFrameThumb	225
8.34.2.2 bTrimThumb	225
8.34.2.3 bVert	225
8.34.2.4 colFrame	225
8.34.2.5 colProgress	225
8.34.2.6 colRemain	225
8.34.2.7 colThumb	226
8.34.2.8 colTick	226
8.34.2.9 colTrim	226
8.34.2.10 nPos	226
8.34.2.11 nPosMax	226
8.34.2.12 nPosMin	226
8.34.2.13 nProgressW	227
8.34.2.14 nRemainW	227
8.34.2.15 nThumbSz	227
8.34.2.16 nTickDiv	227
8.34.2.17 nTickLen	227
8.34.2.18 pfuncXPos	227
8.35 gslc_tsXSelNum Struct Reference	228
8.35.1 Detailed Description	228
8.35.2 Field Documentation	228
8.35.2.1 acElemTxt	228

8.35.2.2	asElem	229
8.35.2.3	asElemRef	229
8.35.2.4	nCounter	229
8.35.2.5	sCollect	229
8.36	gslc_tsXSlider Struct Reference	229
8.36.1	Detailed Description	230
8.36.2	Field Documentation	230
8.36.2.1	bSnapEn	230
8.36.2.2	bTrim	230
8.36.2.3	bVert	231
8.36.2.4	colTick	231
8.36.2.5	colTrim	231
8.36.2.6	nPos	231
8.36.2.7	nPosMax	231
8.36.2.8	nPosMin	231
8.36.2.9	nThumbSz	232
8.36.2.10	nTickDiv	232
8.36.2.11	nTickLen	232
8.36.2.12	pfuncXPos	232
8.37	gslc_tsXSpinner Struct Reference	232
8.37.1	Detailed Description	233
8.37.2	Field Documentation	233
8.37.2.1	acDecr	233
8.37.2.2	acElemTxt	234
8.37.2.3	acIncr	234
8.37.2.4	nCounter	234
8.37.2.5	nFontId	234
8.37.2.6	nIncr	234
8.37.2.7	nMax	234
8.37.2.8	nMin	235

8.37.2.9	pElemRef	235
8.37.2.10	pfuncXInput	235
8.37.2.11	rSubElemBtnDec	235
8.37.2.12	rSubElemBtnInc	235
8.37.2.13	rSubElemTxt	235
8.38	gslc_tsXTemplate Struct Reference	236
8.38.1	Detailed Description	236
8.39	gslc_tsXTextbox Struct Reference	236
8.39.1	Detailed Description	237
8.39.2	Field Documentation	237
8.39.2.1	bScrollEn	237
8.39.2.2	bWrapEn	237
8.39.2.3	nBufCols	237
8.39.2.4	nBufPosX	238
8.39.2.5	nBufPosY	238
8.39.2.6	nBufRows	238
8.39.2.7	nChSizeX	238
8.39.2.8	nChSizeY	238
8.39.2.9	nCurPosX	238
8.39.2.10	nCurPosY	239
8.39.2.11	nMarginX	239
8.39.2.12	nMarginY	239
8.39.2.13	nRedrawRow	239
8.39.2.14	nScrollPos	239
8.39.2.15	nWndCols	239
8.39.2.16	nWndRows	240
8.39.2.17	nWndRowStart	240
8.39.2.18	pBuf	240
8.40	gslc_tsXTogglebtn Struct Reference	240
8.40.1	Detailed Description	241

8.40.2	Field Documentation	241
8.40.2.1	bCircular	241
8.40.2.2	bOn	241
8.40.2.3	colOffState	241
8.40.2.4	colOnState	242
8.40.2.5	colThumb	242
8.40.2.6	nMyPageld	242
8.40.2.7	pfuncUser	242
8.41	gslc_tsXToggleImgbtn Struct Reference	242
8.41.1	Detailed Description	243
8.41.2	Field Documentation	243
8.41.2.1	bOn	243
8.41.2.2	nMyPageld	243
8.41.2.3	pfuncUser	243
8.42	THPoint Class Reference	243
8.42.1	Constructor & Destructor Documentation	244
8.42.1.1	THPoint() [1/2]	244
8.42.1.2	THPoint() [2/2]	244
8.42.2	Member Function Documentation	244
8.42.2.1	operator!=(())	244
8.42.2.2	operator==(())	244
8.42.3	Field Documentation	244
8.42.3.1	x	245
8.42.3.2	y	245
8.42.3.3	z	245
8.43	TouchHandler Class Reference	245
8.43.1	Constructor & Destructor Documentation	246
8.43.1.1	TouchHandler()	246
8.43.2	Member Function Documentation	246
8.43.2.1	begin()	246

8.43.2.2	getPoint()	246
8.43.2.3	scale()	246
8.43.2.4	setCalibration()	246
8.43.2.5	setSize()	247
8.43.2.6	setSwapFlip()	247
8.44	TouchHandler_XPT2046 Class Reference	247
8.44.1	Constructor & Destructor Documentation	248
8.44.1.1	TouchHandler_XPT2046()	248
8.44.2	Member Function Documentation	248
8.44.2.1	begin()	248
8.44.2.2	getPoint()	248
8.44.3	Field Documentation	248
8.44.3.1	spi	249
8.44.3.2	touchDriver	249
<b>9</b>	<b>File Documentation</b>	<b>251</b>
9.1	README.md File Reference	251
9.2	src/elem/XCheckbox.c File Reference	251
9.2.1	Function Documentation	252
9.2.1.1	gslc_ElemXCheckboxCreate()	252
9.2.1.2	gslc_ElemXCheckboxDraw()	253
9.2.1.3	gslc_ElemXCheckboxFindChecked()	253
9.2.1.4	gslc_ElemXCheckboxGetState()	254
9.2.1.5	gslc_ElemXCheckboxSetState()	254
9.2.1.6	gslc_ElemXCheckboxSetStateFunc()	254
9.2.1.7	gslc_ElemXCheckboxSetStateHelp()	255
9.2.1.8	gslc_ElemXCheckboxToggleState()	255
9.2.1.9	gslc_ElemXCheckboxTouch()	255
9.2.2	Variable Documentation	256
9.2.2.1	ERRSTR_NULL	256
9.2.2.2	ERRSTR_PXD_NULL	256

9.3	src/elm/XCheckbox.h File Reference	256
9.3.1	Macro Definition Documentation	258
9.3.1.1	gslc_ElmXCheckboxCreate_P	258
9.3.1.2	GSLC_TYPEX_CHECKBOX	259
9.3.2	Typedef Documentation	259
9.3.2.1	GSLC_CB_XCHECKBOX	259
9.3.3	Enumeration Type Documentation	259
9.3.3.1	gslc_teXCheckboxStyle	259
9.3.4	Function Documentation	260
9.3.4.1	gslc_ElmXCheckboxCreate()	260
9.3.4.2	gslc_ElmXCheckboxDraw()	260
9.3.4.3	gslc_ElmXCheckboxFindChecked()	261
9.3.4.4	gslc_ElmXCheckboxGetState()	261
9.3.4.5	gslc_ElmXCheckboxSetState()	262
9.3.4.6	gslc_ElmXCheckboxSetStateFunc()	262
9.3.4.7	gslc_ElmXCheckboxToggleState()	262
9.3.4.8	gslc_ElmXCheckboxTouch()	263
9.4	src/elm/XGauge.c File Reference	263
9.4.1	Function Documentation	264
9.4.1.1	gslc_ElmXGaugeCreate()	265
9.4.1.2	gslc_ElmXGaugeDraw()	265
9.4.1.3	gslc_ElmXGaugeDrawProgressBar()	266
9.4.1.4	gslc_ElmXGaugeSetFlip()	266
9.4.1.5	gslc_ElmXGaugeSetIndicator()	267
9.4.1.6	gslc_ElmXGaugeSetStyle()	267
9.4.1.7	gslc_ElmXGaugeSetTicks()	268
9.4.1.8	gslc_ElmXGaugeUpdate()	268
9.4.2	Variable Documentation	269
9.4.2.1	ERRSTR_NULL	269
9.4.2.2	ERRSTR_PXD_NULL	269

9.5	src/elem/XGauge.h File Reference	269
9.5.1	Macro Definition Documentation	271
9.5.1.1	gslc_ElemXGaugeCreate_P	271
9.5.1.2	GSLC_TYPEX_GAUGE	272
9.5.2	Enumeration Type Documentation	272
9.5.2.1	gslc_teXGaugeStyle	272
9.5.3	Function Documentation	272
9.5.3.1	gslc_ElemXGaugeCreate()	272
9.5.3.2	gslc_ElemXGaugeDraw()	273
9.5.3.3	gslc_ElemXGaugeDrawProgressBar()	274
9.5.3.4	gslc_ElemXGaugeSetFlip()	274
9.5.3.5	gslc_ElemXGaugeSetIndicator()	275
9.5.3.6	gslc_ElemXGaugeSetStyle()	275
9.5.3.7	gslc_ElemXGaugeSetTicks()	276
9.5.3.8	gslc_ElemXGaugeUpdate()	276
9.6	src/elem/XGlowball.c File Reference	276
9.6.1	Function Documentation	277
9.6.1.1	drawXGlowball()	278
9.6.1.2	drawXGlowballArc()	278
9.6.1.3	drawXGlowballRing()	278
9.6.1.4	gslc_ElemXGlowballCreate()	278
9.6.1.5	gslc_ElemXGlowballDraw()	279
9.6.1.6	gslc_ElemXGlowballSetAngles()	279
9.6.1.7	gslc_ElemXGlowballSetColorBack()	280
9.6.1.8	gslc_ElemXGlowballSetQuality()	280
9.6.1.9	gslc_ElemXGlowballSetVal()	280
9.6.2	Variable Documentation	280
9.6.2.1	ERRSTR_NULL	280
9.6.2.2	ERRSTR_PXD_NULL	280
9.7	src/elem/XGlowball.h File Reference	281

9.7.1	Macro Definition Documentation	282
9.7.1.1	GS�C_TYPEX_GLOW	282
9.7.2	Function Documentation	282
9.7.2.1	drawXGlowball()	282
9.7.2.2	drawXGlowballArc()	282
9.7.2.3	drawXGlowballRing()	283
9.7.2.4	gslc_ElemXGlowballCreate()	283
9.7.2.5	gslc_ElemXGlowballDraw()	283
9.7.2.6	gslc_ElemXGlowballSetAngles()	284
9.7.2.7	gslc_ElemXGlowballSetColorBack()	284
9.7.2.8	gslc_ElemXGlowballSetQuality()	284
9.7.2.9	gslc_ElemXGlowballSetVal()	284
9.8	src/eleŒ/XGraph.c File Reference	285
9.8.1	Function Documentation	286
9.8.1.1	gslc_ElemXGraphAdd()	286
9.8.1.2	gslc_ElemXGraphCreate()	286
9.8.1.3	gslc_ElemXGraphDraw()	287
9.8.1.4	gslc_ElemXGraphReset()	287
9.8.1.5	gslc_ElemXGraphScrollSet()	288
9.8.1.6	gslc_ElemXGraphSetRange()	288
9.8.1.7	gslc_ElemXGraphSetStyle()	288
9.8.2	Variable Documentation	289
9.8.2.1	ERRSTR_NULL	289
9.8.2.2	ERRSTR_PXD_NULL	289
9.9	src/eleŒ/XGraph.h File Reference	289
9.9.1	Macro Definition Documentation	291
9.9.1.1	GS�C_TYPEX_GRAPH	291
9.9.2	Enumeration Type Documentation	291
9.9.2.1	gslc_teXGraphStyle	291
9.9.3	Function Documentation	291



9.9.3.1	<a href="#">gslc_ElemXGraphAdd()</a>	291
9.9.3.2	<a href="#">gslc_ElemXGraphCreate()</a>	292
9.9.3.3	<a href="#">gslc_ElemXGraphDraw()</a>	292
9.9.3.4	<a href="#">gslc_ElemXGraphReset()</a>	293
9.9.3.5	<a href="#">gslc_ElemXGraphScrollSet()</a>	293
9.9.3.6	<a href="#">gslc_ElemXGraphSetRange()</a>	294
9.9.3.7	<a href="#">gslc_ElemXGraphSetStyle()</a>	294
9.10	<a href="#">src/elem/XKeyPad.c File Reference</a>	295
9.10.1	<a href="#">Function Documentation</a>	297
9.10.1.1	<a href="#">gslc_ElemXKeyPadCfgInit()</a>	297
9.10.1.2	<a href="#">gslc_ElemXKeyPadCfgSetButtonSpace()</a>	297
9.10.1.3	<a href="#">gslc_ElemXKeyPadCfgSetButtonSz()</a>	298
9.10.1.4	<a href="#">gslc_ElemXKeyPadCfgSetRoundEn()</a>	298
9.10.1.5	<a href="#">gslc_ElemXKeyPadDataTargetIdGet()</a>	298
9.10.1.6	<a href="#">gslc_ElemXKeyPadDataValGet()</a>	299
9.10.1.7	<a href="#">gslc_ElemXKeyPadInputAsk()</a>	299
9.10.1.8	<a href="#">gslc_ElemXKeyPadInputGet()</a>	300
9.10.1.9	<a href="#">gslc_ElemXKeyPadReset()</a>	300
9.10.1.10	<a href="#">gslc_ElemXKeyPadTargetRefSet()</a>	300
9.10.1.11	<a href="#">gslc_ElemXKeyPadValGet()</a>	301
9.10.1.12	<a href="#">gslc_ElemXKeyPadValSet()</a>	301
9.10.1.13	<a href="#">gslc_ElemXKeyPadValSetCb()</a>	302
9.10.1.14	<a href="#">gslc_XKeyPadAdjustScroll()</a>	302
9.10.1.15	<a href="#">gslc_XKeyPadCreateBase()</a>	302
9.10.1.16	<a href="#">gslc_XKeyPadDraw()</a>	303
9.10.1.17	<a href="#">gslc_XKeyPadDrawKey()</a>	303
9.10.1.18	<a href="#">gslc_XKeyPadDrawLayout()</a>	305
9.10.1.19	<a href="#">gslc_XKeyPadDrawVirtualBtn()</a>	305
9.10.1.20	<a href="#">gslc_XKeyPadDrawVirtualTxt()</a>	306
9.10.1.21	<a href="#">gslc_XKeyPadFocusSetDefault()</a>	306

9.10.1.22	<code>gslc_XKeyPadLayoutSet()</code>	307
9.10.1.23	<code>gslc_XKeyPadLookupId()</code>	307
9.10.1.24	<code>gslc_XKeyPadLookupSpecialId()</code>	308
9.10.1.25	<code>gslc_XKeyPadMapEvent()</code>	308
9.10.1.26	<code>gslc_XKeyPadPendRedrawAddKey()</code>	308
9.10.1.27	<code>gslc_XKeyPadPendRedrawAddTxt()</code>	308
9.10.1.28	<code>gslc_XKeyPadPendRedrawReset()</code>	309
9.10.1.29	<code>gslc_XKeyPadRedrawUpdate()</code>	309
9.10.1.30	<code>gslc_XKeyPadSizeAllGet()</code>	309
9.10.1.31	<code>gslc_XKeyPadSizeGet()</code>	309
9.10.1.32	<code>gslc_XKeyPadTouch()</code>	310
9.10.1.33	<code>gslc_XKeyPadTrackSet()</code>	311
9.10.1.34	<code>gslc_XKeyPadTxtAddCh()</code>	311
9.10.1.35	<code>gslc_XKeyPadTxtAddStr()</code>	311
9.10.1.36	<code>gslc_XKeyPadTxtDelCh()</code>	312
9.10.2	Variable Documentation	312
9.10.2.1	<code>ERRSTR_NULL</code>	313
9.10.2.2	<code>ERRSTR_PXD_NULL</code>	313
9.11	<code>src/elem/XKeyPad.h</code> File Reference	313
9.11.1	Macro Definition Documentation	316
9.11.1.1	<code>DEBUG_XKEYPAD</code>	317
9.11.1.2	<code>GSLC_TYPEX_KEYPAD</code>	317
9.11.1.3	<code>XKEYPAD_BUF_MAX</code>	317
9.11.1.4	<code>XKEYPAD_CB_STATE_CANCEL</code>	317
9.11.1.5	<code>XKEYPAD_CB_STATE_DONE</code>	317
9.11.1.6	<code>XKEYPAD_CB_STATE_UPDATE</code>	317
9.11.1.7	<code>XKEYPAD_CURSOR_CH</code>	317
9.11.1.8	<code>XKEYPAD_CURSOR_ENHANCED</code>	318
9.11.1.9	<code>XKEYPAD_KEY_LEN</code>	318
9.11.1.10	<code>XKEYPAD_REDRAW_ALL</code>	318

9.11.1.11 XKEYPAD_REDRAW_FULL . . . . .	318
9.11.1.12 XKEYPAD_REDRAW_KEY . . . . .	318
9.11.1.13 XKEYPAD_REDRAW_NONE . . . . .	318
9.11.1.14 XKEYPAD_REDRAW_TXT . . . . .	318
9.11.2 Typedef Documentation . . . . .	318
9.11.2.1 GSLC_CB_XKEYPAD_BTN_EVT . . . . .	319
9.11.2.2 GSLC_CB_XKEYPAD_LABEL_GET . . . . .	319
9.11.2.3 GSLC_CB_XKEYPAD_RESET . . . . .	319
9.11.2.4 GSLC_CB_XKEYPAD_SYTLE_GET . . . . .	319
9.11.2.5 GSLC_CB_XKEYPAD_TXT_INIT . . . . .	319
9.11.2.6 gslc_tsKey . . . . .	319
9.11.3 Enumeration Type Documentation . . . . .	319
9.11.3.1 anonymous enum . . . . .	319
9.11.3.2 anonymous enum . . . . .	320
9.11.3.3 gslc_tsXKeyPadAttrib . . . . .	320
9.11.4 Function Documentation . . . . .	320
9.11.4.1 gslc_ElemXKeyPadCfgInit() . . . . .	320
9.11.4.2 gslc_ElemXKeyPadCfgSetButtonSpace() . . . . .	321
9.11.4.3 gslc_ElemXKeyPadCfgSetButtonSz() . . . . .	321
9.11.4.4 gslc_ElemXKeyPadCfgSetRoundEn() . . . . .	322
9.11.4.5 gslc_ElemXKeyPadDataTargetIdGet() . . . . .	322
9.11.4.6 gslc_ElemXKeyPadDataValGet() . . . . .	323
9.11.4.7 gslc_ElemXKeyPadInputAsk() . . . . .	323
9.11.4.8 gslc_ElemXKeyPadInputGet() . . . . .	323
9.11.4.9 gslc_ElemXKeyPadTargetRefSet() . . . . .	324
9.11.4.10 gslc_ElemXKeyPadValGet() . . . . .	324
9.11.4.11 gslc_ElemXKeyPadValSet() . . . . .	326
9.11.4.12 gslc_ElemXKeyPadValSetCb() . . . . .	326
9.11.4.13 gslc_XKeyPadCreateBase() . . . . .	327
9.11.4.14 gslc_XKeyPadDraw() . . . . .	327

9.11.4.15	<a href="#">gslc_XKeyPadDrawKey()</a>	328
9.11.4.16	<a href="#">gslc_XKeyPadDrawVirtualBtn()</a>	328
9.11.4.17	<a href="#">gslc_XKeyPadDrawVirtualTxt()</a>	329
9.11.4.18	<a href="#">gslc_XKeyPadFocusSetDefault()</a>	330
9.11.4.19	<a href="#">gslc_XKeyPadLayoutSet()</a>	330
9.11.4.20	<a href="#">gslc_XKeyPadLookupId()</a>	330
9.11.4.21	<a href="#">gslc_XKeyPadLookupSpecialId()</a>	331
9.11.4.22	<a href="#">gslc_XKeyPadPendRedrawAddKey()</a>	331
9.11.4.23	<a href="#">gslc_XKeyPadPendRedrawAddTxt()</a>	332
9.11.4.24	<a href="#">gslc_XKeyPadPendRedrawReset()</a>	332
9.11.4.25	<a href="#">gslc_XKeyPadRedrawUpdate()</a>	332
9.11.4.26	<a href="#">gslc_XKeyPadSizeAllGet()</a>	332
9.11.4.27	<a href="#">gslc_XKeyPadSizeGet()</a>	333
9.11.4.28	<a href="#">gslc_XKeyPadTouch()</a>	333
9.11.4.29	<a href="#">gslc_XKeyPadTrackSet()</a>	334
9.11.4.30	<a href="#">gslc_XKeyPadTxtAddCh()</a>	334
9.11.4.31	<a href="#">gslc_XKeyPadTxtAddStr()</a>	335
9.11.4.32	<a href="#">gslc_XKeyPadTxtDelCh()</a>	335
9.11.5	<a href="#">Variable Documentation</a>	336
9.11.5.1	<a href="#">RBIT_CTRL</a>	336
9.11.5.2	<a href="#">RBIT_KEYALL</a>	336
9.11.5.3	<a href="#">RBIT_KEYONE</a>	336
9.11.5.4	<a href="#">RBIT_TXT</a>	336
9.12	<a href="#">src/elem/XKeyPad_Alpha-setup.h File Reference</a>	336
9.12.1	<a href="#">Macro Definition Documentation</a>	338
9.12.1.1	<a href="#">XKEYPAD_COL_BASIC_FILL</a>	338
9.12.1.2	<a href="#">XKEYPAD_COL_BASIC_GLOW</a>	338
9.12.1.3	<a href="#">XKEYPAD_COL_DEF_FILL</a>	338
9.12.1.4	<a href="#">XKEYPAD_COL_DEF_FRAME</a>	338
9.12.1.5	<a href="#">XKEYPAD_COL_DEF_GLOW</a>	338

9.12.1.6	XKEYPAD_COL_DEF_TXT . . . . .	339
9.12.1.7	XKEYPAD_COL_DISABLE_FILL . . . . .	339
9.12.1.8	XKEYPAD_COL_DISABLE_TXT . . . . .	339
9.12.1.9	XKEYPAD_COL_ENTER_FILL . . . . .	339
9.12.1.10	XKEYPAD_COL_ENTER_GLOW . . . . .	339
9.12.1.11	XKEYPAD_COL_ESC_FILL . . . . .	339
9.12.1.12	XKEYPAD_COL_ESC_GLOW . . . . .	339
9.12.1.13	XKEYPAD_COL_SCROLL_L_FILL . . . . .	339
9.12.1.14	XKEYPAD_COL_SCROLL_L_GLOW . . . . .	340
9.12.1.15	XKEYPAD_COL_SCROLL_R_FILL . . . . .	340
9.12.1.16	XKEYPAD_COL_SCROLL_R_GLOW . . . . .	340
9.12.1.17	XKEYPAD_COL_SPACE_FILL . . . . .	340
9.12.1.18	XKEYPAD_COL_SPACE_GLOW . . . . .	340
9.12.1.19	XKEYPAD_COL_TEXT_FILL . . . . .	340
9.12.1.20	XKEYPAD_COL_TEXT_GLOW . . . . .	340
9.12.1.21	XKEYPAD_COL_TEXT_TXT . . . . .	340
9.12.1.22	XKEYPAD_DISP_MAX . . . . .	341
9.12.1.23	XKEYPAD_EXTEND_CHAR . . . . .	341
9.12.1.24	XKEYPAD_KEY_H . . . . .	341
9.12.1.25	XKEYPAD_KEY_W . . . . .	341
9.12.1.26	XKEYPAD_LABEL_MAX . . . . .	341
9.12.1.27	XKEYPAD_SPACING_X . . . . .	341
9.12.1.28	XKEYPAD_SPACING_Y . . . . .	341
9.12.2	Enumeration Type Documentation . . . . .	341
9.12.2.1	gslc_teXKeyPadSel . . . . .	341
9.12.3	Variable Documentation . . . . .	342
9.12.3.1	KEYPAD_LAYOUT . . . . .	342
9.12.3.2	KEYPAD_LAYOUTS . . . . .	342
9.12.3.3	KEYPAD_SET_LABEL . . . . .	342
9.12.3.4	KEYPAD_SPECIAL_LABEL . . . . .	342

9.12.3.5	KEYPAD_SPECIAL_SELECT . . . . .	342
9.12.3.6	XKEYPAD_LABEL_SPACE . . . . .	343
9.12.3.7	XKEYPAD_LAYOUT_DEFAULT . . . . .	343
9.13	src/elem/XKeyPad_Alpha.c File Reference . . . . .	343
9.13.1	Function Documentation . . . . .	344
9.13.1.1	gslc_ElemXKeyPadBtnEvt_Alpha() . . . . .	344
9.13.1.2	gslc_ElemXKeyPadCfgInit_Alpha() . . . . .	345
9.13.1.3	gslc_ElemXKeyPadCreate_Alpha() . . . . .	345
9.13.1.4	gslc_ElemXKeyPadLabelGet_Alpha() . . . . .	346
9.13.1.5	gslc_ElemXKeyPadReset_Alpha() . . . . .	346
9.13.1.6	gslc_ElemXKeyPadStyleGet_Alpha() . . . . .	346
9.13.1.7	gslc_ElemXKeyPadTxtInit_Alpha() . . . . .	347
9.13.2	Variable Documentation . . . . .	348
9.13.2.1	ERRSTR_NULL . . . . .	348
9.13.2.2	ERRSTR_PXD_NULL . . . . .	348
9.14	src/elem/XKeyPad_Alpha.h File Reference . . . . .	348
9.14.1	Function Documentation . . . . .	349
9.14.1.1	gslc_ElemXKeyPadBtnEvt_Alpha() . . . . .	350
9.14.1.2	gslc_ElemXKeyPadCfgInit_Alpha() . . . . .	350
9.14.1.3	gslc_ElemXKeyPadCreate_Alpha() . . . . .	350
9.14.1.4	gslc_ElemXKeyPadLabelGet_Alpha() . . . . .	351
9.14.1.5	gslc_ElemXKeyPadReset_Alpha() . . . . .	351
9.14.1.6	gslc_ElemXKeyPadStyleGet_Alpha() . . . . .	352
9.14.1.7	gslc_ElemXKeyPadTxtInit_Alpha() . . . . .	352
9.15	src/elem/XKeyPad_Num-setup.h File Reference . . . . .	353
9.15.1	Macro Definition Documentation . . . . .	355
9.15.1.1	XKEYPAD_COL_BASIC_FILL . . . . .	355
9.15.1.2	XKEYPAD_COL_BASIC_GLOW . . . . .	355
9.15.1.3	XKEYPAD_COL_DECIMAL_FILL . . . . .	355
9.15.1.4	XKEYPAD_COL_DECIMAL_GLOW . . . . .	355

9.15.1.5	XKEYPAD_COL_DEF_FILL . . . . .	356
9.15.1.6	XKEYPAD_COL_DEF_FRAME . . . . .	356
9.15.1.7	XKEYPAD_COL_DEF_GLOW . . . . .	356
9.15.1.8	XKEYPAD_COL_DEF_TXT . . . . .	356
9.15.1.9	XKEYPAD_COL_DISABLE_FILL . . . . .	356
9.15.1.10	XKEYPAD_COL_DISABLE_TXT . . . . .	356
9.15.1.11	XKEYPAD_COL_ENTER_FILL . . . . .	356
9.15.1.12	XKEYPAD_COL_ENTER_GLOW . . . . .	356
9.15.1.13	XKEYPAD_COL_ESC_FILL . . . . .	357
9.15.1.14	XKEYPAD_COL_ESC_GLOW . . . . .	357
9.15.1.15	XKEYPAD_COL_MINUS_FILL . . . . .	357
9.15.1.16	XKEYPAD_COL_MINUS_GLOW . . . . .	357
9.15.1.17	XKEYPAD_COL_SCROLL_L_FILL . . . . .	357
9.15.1.18	XKEYPAD_COL_SCROLL_L_GLOW . . . . .	357
9.15.1.19	XKEYPAD_COL_SCROLL_R_FILL . . . . .	357
9.15.1.20	XKEYPAD_COL_SCROLL_R_GLOW . . . . .	357
9.15.1.21	XKEYPAD_COL_SPACE_FILL . . . . .	358
9.15.1.22	XKEYPAD_COL_SPACE_GLOW . . . . .	358
9.15.1.23	XKEYPAD_COL_TEXT_FILL . . . . .	358
9.15.1.24	XKEYPAD_COL_TEXT_GLOW . . . . .	358
9.15.1.25	XKEYPAD_COL_TEXT_TXT . . . . .	358
9.15.1.26	XKEYPAD_DISP_MAX . . . . .	358
9.15.1.27	XKEYPAD_EXTEND_CHAR . . . . .	358
9.15.1.28	XKEYPAD_KEY_H . . . . .	358
9.15.1.29	XKEYPAD_KEY_W . . . . .	359
9.15.1.30	XKEYPAD_LABEL_MAX . . . . .	359
9.15.1.31	XKEYPAD_SPACING_X . . . . .	359
9.15.1.32	XKEYPAD_SPACING_Y . . . . .	359
9.15.2	Enumeration Type Documentation . . . . .	359
9.15.2.1	anonymous enum . . . . .	359

9.15.2.2	<a href="#">gslc_teXKeyPadSel</a>	359
9.15.3	<a href="#">Variable Documentation</a>	360
9.15.3.1	<a href="#">KEYPAD_LABEL_DECIMAL_PT</a>	360
9.15.3.2	<a href="#">KEYPAD_LABEL_NEGATIVE</a>	360
9.15.3.3	<a href="#">KEYPAD_LAYOUT</a>	360
9.15.3.4	<a href="#">KEYPAD_LAYOUTS</a>	360
9.15.3.5	<a href="#">KEYPAD_SET_LABEL</a>	360
9.15.3.6	<a href="#">KEYPAD_SPECIAL_LABEL</a>	361
9.15.3.7	<a href="#">XKEYPAD_LAYOUT_DEFAULT</a>	361
9.16	<a href="#">src/elem/XKeyPad_Num.c File Reference</a>	361
9.16.1	<a href="#">Function Documentation</a>	362
9.16.1.1	<a href="#">gslc_ElemXKeyPadBtnEvt_Num()</a>	362
9.16.1.2	<a href="#">gslc_ElemXKeyPadCfgInit_Num()</a>	363
9.16.1.3	<a href="#">gslc_ElemXKeyPadCfgSetFloatEn_Num()</a>	363
9.16.1.4	<a href="#">gslc_ElemXKeyPadCfgSetSignEn_Num()</a>	363
9.16.1.5	<a href="#">gslc_ElemXKeyPadCreate_Num()</a>	364
9.16.1.6	<a href="#">gslc_ElemXKeyPadLabelGet_Num()</a>	364
9.16.1.7	<a href="#">gslc_ElemXKeyPadReset_Num()</a>	365
9.16.1.8	<a href="#">gslc_ElemXKeyPadStyleGet_Num()</a>	365
9.16.1.9	<a href="#">gslc_ElemXKeyPadTxtInit_Num()</a>	366
9.16.1.10	<a href="#">gslc_XKeyPadValSetSign_Num()</a>	366
9.16.2	<a href="#">Variable Documentation</a>	367
9.16.2.1	<a href="#">ERRSTR_NULL</a>	367
9.16.2.2	<a href="#">ERRSTR_PXD_NULL</a>	367
9.17	<a href="#">src/elem/XKeyPad_Num.h File Reference</a>	367
9.17.1	<a href="#">Function Documentation</a>	368
9.17.1.1	<a href="#">gslc_ElemXKeyPadBtnEvt_Num()</a>	369
9.17.1.2	<a href="#">gslc_ElemXKeyPadCfgInit_Num()</a>	369
9.17.1.3	<a href="#">gslc_ElemXKeyPadCfgSetFloatEn_Num()</a>	369
9.17.1.4	<a href="#">gslc_ElemXKeyPadCfgSetSignEn_Num()</a>	370



9.17.1.5	<a href="#">gslc_ElemXKeyPadCreate_Num()</a>	370
9.17.1.6	<a href="#">gslc_ElemXKeyPadLabelGet_Num()</a>	371
9.17.1.7	<a href="#">gslc_ElemXKeyPadReset_Num()</a>	371
9.17.1.8	<a href="#">gslc_ElemXKeyPadStyleGet_Num()</a>	372
9.17.1.9	<a href="#">gslc_ElemXKeyPadTxtInit_Num()</a>	372
9.18	<a href="#">src/elem/XListbox.c File Reference</a>	373
9.18.1	<a href="#">Macro Definition Documentation</a>	374
9.18.1.1	<a href="#">XLISTBOX_MAX_STR</a>	375
9.18.2	<a href="#">Function Documentation</a>	375
9.18.2.1	<a href="#">gslc_ElemXListboxAddItem()</a>	375
9.18.2.2	<a href="#">gslc_ElemXListboxCreate()</a>	375
9.18.2.3	<a href="#">gslc_ElemXListboxDeleteItemAt()</a>	376
9.18.2.4	<a href="#">gslc_ElemXListboxDraw()</a>	376
9.18.2.5	<a href="#">gslc_ElemXListboxGetItem()</a>	377
9.18.2.6	<a href="#">gslc_ElemXListboxGetItemAddr()</a>	377
9.18.2.7	<a href="#">gslc_ElemXListboxGetItemCnt()</a>	377
9.18.2.8	<a href="#">gslc_ElemXListboxGetSel()</a>	378
9.18.2.9	<a href="#">gslc_ElemXListboxInsertItemAt()</a>	378
9.18.2.10	<a href="#">gslc_ElemXListboxItemsSetGap()</a>	379
9.18.2.11	<a href="#">gslc_ElemXListboxItemsSetSize()</a>	379
9.18.2.12	<a href="#">gslc_ElemXListboxRecalcSize()</a>	380
9.18.2.13	<a href="#">gslc_ElemXListboxReset()</a>	380
9.18.2.14	<a href="#">gslc_ElemXListboxSetMargin()</a>	380
9.18.2.15	<a href="#">gslc_ElemXListboxSetScrollPos()</a>	381
9.18.2.16	<a href="#">gslc_ElemXListboxSetSel()</a>	381
9.18.2.17	<a href="#">gslc_ElemXListboxSetSelFunc()</a>	382
9.18.2.18	<a href="#">gslc_ElemXListboxSetSize()</a>	382
9.18.2.19	<a href="#">gslc_ElemXListboxTouch()</a>	382
9.18.3	<a href="#">Variable Documentation</a>	383
9.18.3.1	<a href="#">ERRSTR_NULL</a>	383

9.18.3.2	ERRSTR_PXD_NULL	383
9.19	src/elem/XListbox.h File Reference	383
9.19.1	Macro Definition Documentation	385
9.19.1.1	GSLC_TYPEX_LISTBOX	385
9.19.1.2	XLISTBOX_BUF_OH_R	385
9.19.1.3	XLISTBOX_SEL_NONE	385
9.19.1.4	XLISTBOX_SIZE_AUTO	386
9.19.2	Typedef Documentation	386
9.19.2.1	GSLC_CB_XLISTBOX_SEL	386
9.19.3	Function Documentation	386
9.19.3.1	gslc_ElemXListboxAddItem()	386
9.19.3.2	gslc_ElemXListboxCreate()	386
9.19.3.3	gslc_ElemXListboxDeleteItemAt()	387
9.19.3.4	gslc_ElemXListboxDraw()	388
9.19.3.5	gslc_ElemXListboxGetItem()	388
9.19.3.6	gslc_ElemXListboxGetItemCnt()	389
9.19.3.7	gslc_ElemXListboxGetSel()	389
9.19.3.8	gslc_ElemXListboxInsertItemAt()	389
9.19.3.9	gslc_ElemXListboxItemsSetGap()	390
9.19.3.10	gslc_ElemXListboxItemsSetSize()	390
9.19.3.11	gslc_ElemXListboxReset()	391
9.19.3.12	gslc_ElemXListboxSetMargin()	391
9.19.3.13	gslc_ElemXListboxSetScrollPos()	392
9.19.3.14	gslc_ElemXListboxSetSel()	392
9.19.3.15	gslc_ElemXListboxSetSelFunc()	392
9.19.3.16	gslc_ElemXListboxSetSize()	393
9.19.3.17	gslc_ElemXListboxTouch()	393
9.20	src/elem/XProgress.c File Reference	394
9.20.1	Function Documentation	395
9.20.1.1	gslc_ElemXProgressCreate()	395

9.20.1.2	<a href="#">gslc_ElemXProgressDraw()</a>	396
9.20.1.3	<a href="#">gslc_ElemXProgressDrawHelp()</a>	396
9.20.1.4	<a href="#">gslc_ElemXProgressSetFlip()</a>	397
9.20.1.5	<a href="#">gslc_ElemXProgressSetGaugeCol()</a>	397
9.20.1.6	<a href="#">gslc_ElemXProgressSetVal()</a>	398
9.20.2	Variable Documentation	398
9.20.2.1	<a href="#">ERRSTR_NULL</a>	398
9.20.2.2	<a href="#">ERRSTR_PXD_NULL</a>	398
9.21	<a href="#">src/elem/XProgress.h</a> File Reference	398
9.21.1	Macro Definition Documentation	399
9.21.1.1	<a href="#">gslc_ElemXProgressCreate_P</a>	400
9.21.1.2	<a href="#">GSLC_TYPEX_PROGRESS</a>	400
9.21.2	Function Documentation	400
9.21.2.1	<a href="#">gslc_ElemXProgressCreate()</a>	401
9.21.2.2	<a href="#">gslc_ElemXProgressDraw()</a>	401
9.21.2.3	<a href="#">gslc_ElemXProgressDrawHelp()</a>	402
9.21.2.4	<a href="#">gslc_ElemXProgressSetFlip()</a>	402
9.21.2.5	<a href="#">gslc_ElemXProgressSetGaugeCol()</a>	403
9.21.2.6	<a href="#">gslc_ElemXProgressSetVal()</a>	403
9.22	<a href="#">src/elem/XRadial.c</a> File Reference	404
9.22.1	Function Documentation	405
9.22.1.1	<a href="#">gslc_ElemXRadialCreate()</a>	405
9.22.1.2	<a href="#">gslc_ElemXRadialDraw()</a>	405
9.22.1.3	<a href="#">gslc_ElemXRadialDrawRadial()</a>	406
9.22.1.4	<a href="#">gslc_ElemXRadialDrawRadialHelp()</a>	406
9.22.1.5	<a href="#">gslc_ElemXRadialSetFlip()</a>	407
9.22.1.6	<a href="#">gslc_ElemXRadialSetIndicator()</a>	407
9.22.1.7	<a href="#">gslc_ElemXRadialSetTicks()</a>	408
9.22.1.8	<a href="#">gslc_ElemXRadialSetVal()</a>	408
9.22.2	Variable Documentation	409

9.22.2.1	ERRSTR_NULL	409
9.22.2.2	ERRSTR_PXD_NULL	409
9.23	src/elem/XRadial.h File Reference	409
9.23.1	Macro Definition Documentation	411
9.23.1.1	gslc_ElemXRadialCreate_P	411
9.23.1.2	GSLC_TYPEX_RADIAL	411
9.23.2	Function Documentation	412
9.23.2.1	gslc_ElemXRadialCreate()	412
9.23.2.2	gslc_ElemXRadialDraw()	412
9.23.2.3	gslc_ElemXRadialDrawRadial()	413
9.23.2.4	gslc_ElemXRadialSetFlip()	413
9.23.2.5	gslc_ElemXRadialSetIndicator()	414
9.23.2.6	gslc_ElemXRadialSetTicks()	414
9.23.2.7	gslc_ElemXRadialSetVal()	415
9.24	src/elem/XRamp.c File Reference	415
9.24.1	Function Documentation	416
9.24.1.1	gslc_ElemXRampCreate()	416
9.24.1.2	gslc_ElemXRampDraw()	417
9.24.1.3	gslc_ElemXRampDrawHelp()	418
9.24.1.4	gslc_ElemXRampSetVal()	418
9.24.2	Variable Documentation	419
9.24.2.1	ERRSTR_NULL	419
9.24.2.2	ERRSTR_PXD_NULL	419
9.25	src/elem/XRamp.h File Reference	419
9.25.1	Macro Definition Documentation	420
9.25.1.1	gslc_ElemXRampCreate_P	420
9.25.1.2	GSLC_TYPEX_RAMP	421
9.25.2	Function Documentation	421
9.25.2.1	gslc_ElemXRampCreate()	421
9.25.2.2	gslc_ElemXRampDraw()	422

9.25.2.3	<a href="#">gslc_ElemXRampDrawHelp()</a>	422
9.25.2.4	<a href="#">gslc_ElemXRampSetVal()</a>	423
9.26	<a href="#">src/elem/XRingGauge.c File Reference</a>	423
9.26.1	<a href="#">Function Documentation</a>	425
9.26.1.1	<a href="#">gslc_ElemXRingGaugeCreate()</a>	425
9.26.1.2	<a href="#">gslc_ElemXRingGaugeDraw()</a>	425
9.26.1.3	<a href="#">gslc_ElemXRingGaugeSetAngleRange()</a>	426
9.26.1.4	<a href="#">gslc_ElemXRingGaugeSetColorActiveFlat()</a>	427
9.26.1.5	<a href="#">gslc_ElemXRingGaugeSetColorActiveGradient()</a>	427
9.26.1.6	<a href="#">gslc_ElemXRingGaugeSetColorInactive()</a>	427
9.26.1.7	<a href="#">gslc_ElemXRingGaugeSetQuality()</a>	428
9.26.1.8	<a href="#">gslc_ElemXRingGaugeSetThickness()</a>	428
9.26.1.9	<a href="#">gslc_ElemXRingGaugeSetVal()</a>	429
9.26.1.10	<a href="#">gslc_ElemXRingGaugeSetValRange()</a>	430
9.26.2	<a href="#">Variable Documentation</a>	430
9.26.2.1	<a href="#">ERRSTR_NULL</a>	430
9.26.2.2	<a href="#">ERRSTR_PXD_NULL</a>	430
9.27	<a href="#">src/elem/XRingGauge.h File Reference</a>	431
9.27.1	<a href="#">Macro Definition Documentation</a>	432
9.27.1.1	<a href="#">GSLC_TYPEX_RING</a>	432
9.27.1.2	<a href="#">XRING_STR_MAX</a>	432
9.27.2	<a href="#">Function Documentation</a>	432
9.27.2.1	<a href="#">gslc_ElemXRingGaugeCreate()</a>	432
9.27.2.2	<a href="#">gslc_ElemXRingGaugeDraw()</a>	433
9.27.2.3	<a href="#">gslc_ElemXRingGaugeSetAngleRange()</a>	433
9.27.2.4	<a href="#">gslc_ElemXRingGaugeSetColorActiveFlat()</a>	434
9.27.2.5	<a href="#">gslc_ElemXRingGaugeSetColorActiveGradient()</a>	435
9.27.2.6	<a href="#">gslc_ElemXRingGaugeSetColorInactive()</a>	435
9.27.2.7	<a href="#">gslc_ElemXRingGaugeSetQuality()</a>	436
9.27.2.8	<a href="#">gslc_ElemXRingGaugeSetThickness()</a>	436

9.27.2.9	<code>gslc_ElemXRingGaugeSetVal()</code>	437
9.27.2.10	<code>gslc_ElemXRingGaugeSetValRange()</code>	437
9.28	<code>src/elem/XSeekBar.c</code> File Reference	438
9.28.1	Function Documentation	439
9.28.1.1	<code>gslc_ElemXSeekBarCreate()</code>	439
9.28.1.2	<code>gslc_ElemXSeekBarDraw()</code>	440
9.28.1.3	<code>gslc_ElemXSeekBarGetPos()</code>	440
9.28.1.4	<code>gslc_ElemXSeekBarSetPos()</code>	440
9.28.1.5	<code>gslc_ElemXSeekBarSetPosFunc()</code>	441
9.28.1.6	<code>gslc_ElemXSeekBarSetStyle()</code>	441
9.28.1.7	<code>gslc_ElemXSeekBarTouch()</code>	442
9.28.2	Variable Documentation	442
9.28.2.1	<code>ERRSTR_NULL</code>	443
9.28.2.2	<code>ERRSTR_PXD_NULL</code>	443
9.29	<code>src/elem/XSeekBar.h</code> File Reference	443
9.29.1	Macro Definition Documentation	444
9.29.1.1	<code>gslc_ElemXSeekBarCreate_P</code>	445
9.29.1.2	<code>GSLC_TYPEX_SEEKBAR</code>	446
9.29.2	Typedef Documentation	446
9.29.2.1	<code>GSLC_CB_XSEEKBAR_POS</code>	446
9.29.3	Function Documentation	446
9.29.3.1	<code>gslc_ElemXSeekBarCreate()</code>	446
9.29.3.2	<code>gslc_ElemXSeekBarDraw()</code>	447
9.29.3.3	<code>gslc_ElemXSeekBarGetPos()</code>	448
9.29.3.4	<code>gslc_ElemXSeekBarSetPos()</code>	448
9.29.3.5	<code>gslc_ElemXSeekBarSetPosFunc()</code>	448
9.29.3.6	<code>gslc_ElemXSeekBarSetStyle()</code>	449
9.29.3.7	<code>gslc_ElemXSeekBarTouch()</code>	449
9.30	<code>src/elem/XSelNum.c</code> File Reference	450
9.30.1	Function Documentation	451

9.30.1.1	<a href="#">gslc_ElemXSelNumClick()</a>	451
9.30.1.2	<a href="#">gslc_ElemXSelNumCreate()</a>	452
9.30.1.3	<a href="#">gslc_ElemXSelNumDraw()</a>	452
9.30.1.4	<a href="#">gslc_ElemXSelNumGetCounter()</a>	453
9.30.1.5	<a href="#">gslc_ElemXSelNumSetCounter()</a>	453
9.30.1.6	<a href="#">gslc_ElemXSelNumTouch()</a>	453
9.30.2	<a href="#">Variable Documentation</a>	454
9.30.2.1	<a href="#">ERRSTR_NULL</a>	454
9.30.2.2	<a href="#">ERRSTR_PXD_NULL</a>	454
9.30.2.3	<a href="#">SELNUM_ID_BTN_DEC</a>	454
9.30.2.4	<a href="#">SELNUM_ID_BTN_INC</a>	454
9.30.2.5	<a href="#">SELNUM_ID_TXT</a>	455
9.31	<a href="#">src/elem/XSelNum.h File Reference</a>	455
9.31.1	<a href="#">Macro Definition Documentation</a>	456
9.31.1.1	<a href="#">GSLC_TYPEX_SELNUM</a>	456
9.31.1.2	<a href="#">SELNUM_STR_LEN</a>	456
9.31.2	<a href="#">Function Documentation</a>	456
9.31.2.1	<a href="#">gslc_ElemXSelNumClick()</a>	456
9.31.2.2	<a href="#">gslc_ElemXSelNumCreate()</a>	457
9.31.2.3	<a href="#">gslc_ElemXSelNumDraw()</a>	457
9.31.2.4	<a href="#">gslc_ElemXSelNumGetCounter()</a>	458
9.31.2.5	<a href="#">gslc_ElemXSelNumSetCounter()</a>	458
9.31.2.6	<a href="#">gslc_ElemXSelNumTouch()</a>	459
9.32	<a href="#">src/elem/XSlider.c File Reference</a>	459
9.32.1	<a href="#">Function Documentation</a>	460
9.32.1.1	<a href="#">gslc_ElemXSliderCreate()</a>	461
9.32.1.2	<a href="#">gslc_ElemXSliderDraw()</a>	461
9.32.1.3	<a href="#">gslc_ElemXSliderGetPos()</a>	462
9.32.1.4	<a href="#">gslc_ElemXSliderSetPos()</a>	462
9.32.1.5	<a href="#">gslc_ElemXSliderSetPosFunc()</a>	463

9.32.1.6	<code>gslc_ElemXSliderSetSnapEn()</code>	463
9.32.1.7	<code>gslc_ElemXSliderSetStyle()</code>	463
9.32.1.8	<code>gslc_ElemXSliderTouch()</code>	464
9.32.2	Variable Documentation	465
9.32.2.1	<code>ERRSTR_NULL</code>	465
9.32.2.2	<code>ERRSTR_PXD_NULL</code>	465
9.33	<code>src/elem/XSlider.h</code> File Reference	465
9.33.1	Macro Definition Documentation	466
9.33.1.1	<code>gslc_ElemXSliderCreate_P</code>	467
9.33.1.2	<code>GSLC_TYPEX_SLIDER</code>	467
9.33.2	Typedef Documentation	468
9.33.2.1	<code>GSLC_CB_XSLIDER_POS</code>	468
9.33.3	Function Documentation	468
9.33.3.1	<code>gslc_ElemXSliderCreate()</code>	468
9.33.3.2	<code>gslc_ElemXSliderDraw()</code>	469
9.33.3.3	<code>gslc_ElemXSliderGetPos()</code>	469
9.33.3.4	<code>gslc_ElemXSliderSetPos()</code>	469
9.33.3.5	<code>gslc_ElemXSliderSetPosFunc()</code>	470
9.33.3.6	<code>gslc_ElemXSliderSetSnapEn()</code>	470
9.33.3.7	<code>gslc_ElemXSliderSetStyle()</code>	471
9.33.3.8	<code>gslc_ElemXSliderTouch()</code>	471
9.34	<code>src/elem/XSpinner.c</code> File Reference	472
9.34.1	Detailed Description	473
9.34.2	Function Documentation	473
9.34.2.1	<code>gslc_ElemXSpinnerCreate()</code>	473
9.34.2.2	<code>gslc_ElemXSpinnerDraw()</code>	474
9.34.2.3	<code>gslc_ElemXSpinnerGetCounter()</code>	474
9.34.2.4	<code>gslc_ElemXSpinnerSetChars()</code>	475
9.34.2.5	<code>gslc_ElemXSpinnerSetCounter()</code>	475
9.34.2.6	<code>gslc_ElemXSpinnerTouch()</code>	476



9.34.2.7	<a href="#">gslc_XSpinnerDrawVirtualBtn()</a>	476
9.34.2.8	<a href="#">gslc_XSpinnerDrawVirtualTxt()</a>	477
9.34.2.9	<a href="#">gslc_XSpinnerMapEvent()</a>	477
9.34.3	<a href="#">Variable Documentation</a>	477
9.34.3.1	<a href="#">ERRSTR_NULL</a>	477
9.34.3.2	<a href="#">ERRSTR_PXD_NULL</a>	477
9.34.3.3	<a href="#">SPINNER_ID_BTN_DEC</a>	478
9.34.3.4	<a href="#">SPINNER_ID_BTN_INC</a>	478
9.34.3.5	<a href="#">SPINNER_ID_TXT</a>	478
9.35	<a href="#">src/elem/XSpinner.h File Reference</a>	478
9.35.1	<a href="#">Detailed Description</a>	479
9.35.2	<a href="#">Macro Definition Documentation</a>	479
9.35.2.1	<a href="#">GSLC_TYPEX_SPINNER</a>	479
9.35.2.2	<a href="#">XSPINNER_CB_STATE_UPDATE</a>	480
9.35.2.3	<a href="#">XSPINNER_COMP_CNT</a>	480
9.35.2.4	<a href="#">XSPINNER_STR_LEN</a>	480
9.35.3	<a href="#">Function Documentation</a>	480
9.35.3.1	<a href="#">gslc_ElemXSpinnerClick()</a>	480
9.35.3.2	<a href="#">gslc_ElemXSpinnerCreate()</a>	481
9.35.3.3	<a href="#">gslc_ElemXSpinnerDraw()</a>	481
9.35.3.4	<a href="#">gslc_ElemXSpinnerGetCounter()</a>	482
9.35.3.5	<a href="#">gslc_ElemXSpinnerSetChars()</a>	482
9.35.3.6	<a href="#">gslc_ElemXSpinnerSetCounter()</a>	483
9.35.3.7	<a href="#">gslc_ElemXSpinnerTouch()</a>	483
9.36	<a href="#">src/elem/XTemplate.c File Reference</a>	484
9.36.1	<a href="#">Function Documentation</a>	484
9.36.1.1	<a href="#">gslc_ElemXTemplateCreate()</a>	485
9.36.1.2	<a href="#">gslc_ElemXTemplateDraw()</a>	485
9.36.1.3	<a href="#">gslc_ElemXTemplateTouch()</a>	486
9.36.2	<a href="#">Variable Documentation</a>	486

9.36.2.1	ERRSTR_NULL	486
9.36.2.2	ERRSTR_PXD_NULL	486
9.37	src/elem/XTemplate.h File Reference	487
9.37.1	Macro Definition Documentation	488
9.37.1.1	GSLC_TYPEX_TEMPLATE	488
9.37.2	Function Documentation	488
9.37.2.1	gslc_ElemXTemplateCreate()	488
9.37.2.2	gslc_ElemXTemplateDraw()	488
9.37.2.3	gslc_ElemXTemplateTouch()	489
9.38	src/elem/XTextbox.c File Reference	489
9.38.1	Function Documentation	490
9.38.1.1	gslc_ElemXTextboxAdd()	491
9.38.1.2	gslc_ElemXTextboxBufAdd()	491
9.38.1.3	gslc_ElemXTextboxColReset()	491
9.38.1.4	gslc_ElemXTextboxColSet()	492
9.38.1.5	gslc_ElemXTextboxCreate()	492
9.38.1.6	gslc_ElemXTextboxDraw()	493
9.38.1.7	gslc_ElemXTextboxLineWrAdv()	493
9.38.1.8	gslc_ElemXTextboxReset()	494
9.38.1.9	gslc_ElemXTextboxScrollSet()	494
9.38.1.10	gslc_ElemXTextboxWrapSet()	494
9.38.2	Variable Documentation	495
9.38.2.1	ERRSTR_NULL	495
9.38.2.2	ERRSTR_PXD_NULL	495
9.39	src/elem/XTextbox.h File Reference	495
9.39.1	Macro Definition Documentation	497
9.39.1.1	GSLC_TYPEX_TEXTBOX	497
9.39.1.2	GSLC_XTEXTBOX_CODE_COL_RESET	497
9.39.1.3	GSLC_XTEXTBOX_CODE_COL_SET	497
9.39.1.4	XTEXTBOX_REDRAW_ALL	497

9.39.1.5	XTEXTBOX_REDRAW_NONE	497
9.39.2	Function Documentation	497
9.39.2.1	gslc_ElemXTextboxAdd()	497
9.39.2.2	gslc_ElemXTextboxColReset()	498
9.39.2.3	gslc_ElemXTextboxColSet()	498
9.39.2.4	gslc_ElemXTextboxCreate()	499
9.39.2.5	gslc_ElemXTextboxDraw()	499
9.39.2.6	gslc_ElemXTextboxReset()	500
9.39.2.7	gslc_ElemXTextboxScrollSet()	500
9.39.2.8	gslc_ElemXTextboxWrapSet()	501
9.40	src/ele/XTogglebtn.c File Reference	501
9.40.1	Function Documentation	502
9.40.1.1	gslc_ElemXTogglebtnCreate()	503
9.40.1.2	gslc_ElemXTogglebtnDraw()	503
9.40.1.3	gslc_ElemXTogglebtnDrawCircularHelp()	504
9.40.1.4	gslc_ElemXTogglebtnDrawRectangularHelp()	504
9.40.1.5	gslc_ElemXTogglebtnFindSelected()	504
9.40.1.6	gslc_ElemXTogglebtnGetState()	505
9.40.1.7	gslc_ElemXTogglebtnSetState()	505
9.40.1.8	gslc_ElemXTogglebtnSetStateHelp()	505
9.40.1.9	gslc_ElemXTogglebtnToggleState()	506
9.40.1.10	gslc_ElemXTogglebtnTouch()	506
9.40.2	Variable Documentation	506
9.40.2.1	ERRSTR_NULL	507
9.40.2.2	ERRSTR_PXD_NULL	507
9.41	src/ele/XTogglebtn.h File Reference	507
9.41.1	Macro Definition Documentation	508
9.41.1.1	gslc_ElemXTogglebtnCreate_P	508
9.41.1.2	GSLC_TYPEX_TOGGLEBTN	509
9.41.2	Function Documentation	509

9.41.2.1	<a href="#">gslc_ElemXTogglebtnCreate()</a>	509
9.41.2.2	<a href="#">gslc_ElemXTogglebtnDraw()</a>	510
9.41.2.3	<a href="#">gslc_ElemXTogglebtnFindSelected()</a>	510
9.41.2.4	<a href="#">gslc_ElemXTogglebtnGetState()</a>	512
9.41.2.5	<a href="#">gslc_ElemXTogglebtnSetState()</a>	512
9.41.2.6	<a href="#">gslc_ElemXTogglebtnToggleState()</a>	513
9.41.2.7	<a href="#">gslc_ElemXTogglebtnTouch()</a>	513
9.42	<a href="#">src/elem/XToggleImgbtn.c File Reference</a>	514
9.42.1	<a href="#">Function Documentation</a>	515
9.42.1.1	<a href="#">gslc_ElemXToggleImgbtnCreate()</a>	515
9.42.1.2	<a href="#">gslc_ElemXToggleImgbtnDraw()</a>	515
9.42.1.3	<a href="#">gslc_ElemXToggleImgbtnFindSelected()</a>	516
9.42.1.4	<a href="#">gslc_ElemXToggleImgbtnGetState()</a>	516
9.42.1.5	<a href="#">gslc_ElemXToggleImgbtnSetState()</a>	517
9.42.1.6	<a href="#">gslc_ElemXToggleImgbtnSetStateHelp()</a>	517
9.42.1.7	<a href="#">gslc_ElemXToggleImgbtnToggleState()</a>	517
9.42.1.8	<a href="#">gslc_ElemXToggleImgbtnTouch()</a>	518
9.42.2	<a href="#">Variable Documentation</a>	518
9.42.2.1	<a href="#">ERRSTR_NULL</a>	518
9.42.2.2	<a href="#">ERRSTR_PXD_NULL</a>	518
9.43	<a href="#">src/elem/XToggleImgbtn.h File Reference</a>	519
9.43.1	<a href="#">Macro Definition Documentation</a>	520
9.43.1.1	<a href="#">gslc_ElemXToggleImgbtnCreate_P</a>	520
9.43.1.2	<a href="#">GSLC_TYPEX_TOGGLEIMGBTN</a>	521
9.43.2	<a href="#">Function Documentation</a>	521
9.43.2.1	<a href="#">gslc_ElemXToggleImgbtnCreate()</a>	521
9.43.2.2	<a href="#">gslc_ElemXToggleImgbtnDraw()</a>	522
9.43.2.3	<a href="#">gslc_ElemXToggleImgbtnFindSelected()</a>	522
9.43.2.4	<a href="#">gslc_ElemXToggleImgbtnGetState()</a>	522
9.43.2.5	<a href="#">gslc_ElemXToggleImgbtnSetState()</a>	523

9.43.2.6	<a href="#">gslc_ElemXToggleImgbtnToggleState()</a>	523
9.43.2.7	<a href="#">gslc_ElemXToggleImgbtnTouch()</a>	524
9.44	<a href="#">src/GUIslice.c File Reference</a>	524
9.44.1	<a href="#">Enumeration Type Documentation</a>	533
9.44.1.1	<a href="#">gslc_teDebugPrintState</a>	533
9.44.2	<a href="#">Function Documentation</a>	534
9.44.2.1	<a href="#">gslc_DrawFillSectorBase()</a>	534
9.44.2.2	<a href="#">gslc_ElemCanFocus()</a>	534
9.44.2.3	<a href="#">gslc_ElemGetEditEn()</a>	534
9.44.2.4	<a href="#">gslc_FontSetBase()</a>	534
9.44.2.5	<a href="#">gslc_ImgRefEqual()</a>	535
9.44.2.6	<a href="#">gslc_OrderCoord()</a>	535
9.44.2.7	<a href="#">gslc_SwapCoords()</a>	535
9.44.3	<a href="#">Variable Documentation</a>	535
9.44.3.1	<a href="#">ERRSTR_NULL</a>	535
9.44.3.2	<a href="#">ERRSTR_PXD_NULL</a>	535
9.44.3.3	<a href="#">g_pfDebugOut</a>	535
9.44.3.4	<a href="#">m_nLUTSinF0X16</a>	536
9.45	<a href="#">src/GUIslice.h File Reference</a>	536
9.45.1	<a href="#">Macro Definition Documentation</a>	551
9.45.1.1	<a href="#">GSLC_2PI</a>	551
9.45.1.2	<a href="#">GSLC_ALIGN_BOT_LEFT</a>	551
9.45.1.3	<a href="#">GSLC_ALIGN_BOT_MID</a>	551
9.45.1.4	<a href="#">GSLC_ALIGN_BOT_RIGHT</a>	552
9.45.1.5	<a href="#">GSLC_ALIGN_MID_LEFT</a>	552
9.45.1.6	<a href="#">GSLC_ALIGN_MID_MID</a>	552
9.45.1.7	<a href="#">GSLC_ALIGN_MID_RIGHT</a>	552
9.45.1.8	<a href="#">GSLC_ALIGN_TOP_LEFT</a>	552
9.45.1.9	<a href="#">GSLC_ALIGN_TOP_MID</a>	552
9.45.1.10	<a href="#">GSLC_ALIGN_TOP_RIGHT</a>	553

9.45.1.11 GSLC_ALIGNH_LEFT . . . . .	553
9.45.1.12 GSLC_ALIGNH_MID . . . . .	553
9.45.1.13 GSLC_ALIGNH_RIGHT . . . . .	553
9.45.1.14 GSLC_ALIGNV_BOT . . . . .	553
9.45.1.15 GSLC_ALIGNV_MID . . . . .	553
9.45.1.16 GSLC_ALIGNV_TOP . . . . .	554
9.45.1.17 GSLC_COL_BLACK . . . . .	554
9.45.1.18 GSLC_COL_BLUE . . . . .	554
9.45.1.19 GSLC_COL_BLUE_DK1 . . . . .	554
9.45.1.20 GSLC_COL_BLUE_DK2 . . . . .	554
9.45.1.21 GSLC_COL_BLUE_DK3 . . . . .	554
9.45.1.22 GSLC_COL_BLUE_DK4 . . . . .	555
9.45.1.23 GSLC_COL_BLUE_LT1 . . . . .	555
9.45.1.24 GSLC_COL_BLUE_LT2 . . . . .	555
9.45.1.25 GSLC_COL_BLUE_LT3 . . . . .	555
9.45.1.26 GSLC_COL_BLUE_LT4 . . . . .	555
9.45.1.27 GSLC_COL_BROWN . . . . .	555
9.45.1.28 GSLC_COL_CYAN . . . . .	556
9.45.1.29 GSLC_COL_GRAY . . . . .	556
9.45.1.30 GSLC_COL_GRAY_DK1 . . . . .	556
9.45.1.31 GSLC_COL_GRAY_DK2 . . . . .	556
9.45.1.32 GSLC_COL_GRAY_DK3 . . . . .	556
9.45.1.33 GSLC_COL_GRAY_DK4 . . . . .	556
9.45.1.34 GSLC_COL_GRAY_LT1 . . . . .	557
9.45.1.35 GSLC_COL_GRAY_LT2 . . . . .	557
9.45.1.36 GSLC_COL_GRAY_LT3 . . . . .	557
9.45.1.37 GSLC_COL_GRAY_LT4 . . . . .	557
9.45.1.38 GSLC_COL_GREEN . . . . .	557
9.45.1.39 GSLC_COL_GREEN_DK1 . . . . .	557
9.45.1.40 GSLC_COL_GREEN_DK2 . . . . .	558

9.45.1.41 GSLC_COL_GREEN_DK3 . . . . .	558
9.45.1.42 GSLC_COL_GREEN_DK4 . . . . .	558
9.45.1.43 GSLC_COL_GREEN_LT1 . . . . .	558
9.45.1.44 GSLC_COL_GREEN_LT2 . . . . .	558
9.45.1.45 GSLC_COL_GREEN_LT3 . . . . .	558
9.45.1.46 GSLC_COL_GREEN_LT4 . . . . .	559
9.45.1.47 GSLC_COL_MAGENTA . . . . .	559
9.45.1.48 GSLC_COL_ORANGE . . . . .	559
9.45.1.49 GSLC_COL_PURPLE . . . . .	559
9.45.1.50 GSLC_COL_RED . . . . .	559
9.45.1.51 GSLC_COL_RED_DK1 . . . . .	559
9.45.1.52 GSLC_COL_RED_DK2 . . . . .	560
9.45.1.53 GSLC_COL_RED_DK3 . . . . .	560
9.45.1.54 GSLC_COL_RED_DK4 . . . . .	560
9.45.1.55 GSLC_COL_RED_LT1 . . . . .	560
9.45.1.56 GSLC_COL_RED_LT2 . . . . .	560
9.45.1.57 GSLC_COL_RED_LT3 . . . . .	560
9.45.1.58 GSLC_COL_RED_LT4 . . . . .	561
9.45.1.59 GSLC_COL_TEAL . . . . .	561
9.45.1.60 GSLC_COL_WHITE . . . . .	561
9.45.1.61 GSLC_COL_YELLOW . . . . .	561
9.45.1.62 GSLC_COL_YELLOW_DK . . . . .	561
9.45.1.63 GSLC_COLMONO_BLACK . . . . .	561
9.45.1.64 GSLC_COLMONO_WHITE . . . . .	562
9.45.1.65 GSLC_ELEM_FEA_CLICK_EN . . . . .	562
9.45.1.66 GSLC_ELEM_FEA_EDIT_EN . . . . .	562
9.45.1.67 GSLC_ELEM_FEA_FILL_EN . . . . .	562
9.45.1.68 GSLC_ELEM_FEA_FOCUS_EN . . . . .	562
9.45.1.69 GSLC_ELEM_FEA_FRAME_EN . . . . .	562
9.45.1.70 GSLC_ELEM_FEA_GLOW_EN . . . . .	563

9.45.1.71	GSLC_ELEM_FEA_NONE	563
9.45.1.72	GSLC_ELEM_FEA_NOSHRINK	563
9.45.1.73	GSLC_ELEM_FEA_ROUND_EN	563
9.45.1.74	GSLC_ELEM_FEA_VALID	563
9.45.1.75	GSLC_ELEMREF_DEFAULT	563
9.45.1.76	GSLC_FEATURE_FOCUS_ON_TOUCH	564
9.45.1.77	GSLC_MAX	564
9.45.1.78	GSLC_MIN	564
9.45.1.79	GSLC_PMEM	564
9.45.2	Typedef Documentation	564
9.45.2.1	GSLC_CB_DEBUG_OUT	564
9.45.2.2	GSLC_CB_DRAW	564
9.45.2.3	GSLC_CB_EVENT	565
9.45.2.4	GSLC_CB_INPUT	565
9.45.2.5	GSLC_CB_PIN_POLL	565
9.45.2.6	GSLC_CB_TICK	565
9.45.2.7	GSLC_CB_TOUCH	565
9.45.2.8	gslc_tsColor	565
9.45.2.9	gslc_tsElem	566
9.45.2.10	gslc_tsEvent	566
9.45.2.11	gslc_tsEventTouch	566
9.45.2.12	gslc_tsPt	566
9.45.2.13	gslc_tsRect	566
9.45.2.14	gslc_tsRectState	567
9.45.3	Enumeration Type Documentation	567
9.45.3.1	gslc_teAction	567
9.45.3.2	gslc_teElemId	567
9.45.3.3	gslc_teElemInd	568
9.45.3.4	gslc_teElemRefFlags	568
9.45.3.5	gslc_teEventSubType	569



9.45.3.6	<a href="#">gslc_teEventType</a>	569
9.45.3.7	<a href="#">gslc_teFontId</a>	569
9.45.3.8	<a href="#">gslc_teFontRefMode</a>	570
9.45.3.9	<a href="#">gslc_teFontRefType</a>	570
9.45.3.10	<a href="#">gslc_teGroupId</a>	571
9.45.3.11	<a href="#">gslc_telmgRefFlags</a>	571
9.45.3.12	<a href="#">gslc_telnitStat</a>	571
9.45.3.13	<a href="#">gslc_telinputMode</a>	572
9.45.3.14	<a href="#">gslc_telinputRawEvent</a>	572
9.45.3.15	<a href="#">gslc_tePageId</a>	572
9.45.3.16	<a href="#">gslc_tePin</a>	573
9.45.3.17	<a href="#">gslc_teRedrawType</a>	573
9.45.3.18	<a href="#">gslc_teStackPage</a>	574
9.45.3.19	<a href="#">gslc_teTouch</a>	574
9.45.3.20	<a href="#">gslc_teTxtFlags</a>	574
9.45.3.21	<a href="#">gslc_teTypeCore</a>	575
9.45.4	<a href="#">Variable Documentation</a>	575
9.45.4.1	<a href="#">g_pfDebugOut</a>	576
9.46	<a href="#">src/GUISlice_config.h File Reference</a>	576
9.47	<a href="#">src/GUISlice_drv.h File Reference</a>	576
9.48	<a href="#">src/GUISlice_drv_adagfx.cpp File Reference</a>	576
9.49	<a href="#">src/GUISlice_drv_adagfx.h File Reference</a>	577
9.49.1	<a href="#">Detailed Description</a>	579
9.49.2	<a href="#">Macro Definition Documentation</a>	579
9.49.2.1	<a href="#">DRV_HAS_DRAW_BMP_MEM</a>	580
9.49.2.2	<a href="#">DRV_HAS_DRAW_CIRCLE_FILL</a>	580
9.49.2.3	<a href="#">DRV_HAS_DRAW_CIRCLE_FRAME</a>	580
9.49.2.4	<a href="#">DRV_HAS_DRAW_LINE</a>	580
9.49.2.5	<a href="#">DRV_HAS_DRAW_POINT</a>	580
9.49.2.6	<a href="#">DRV_HAS_DRAW_POINTS</a>	580

9.49.2.7	DRV_HAS_DRAW_RECT_FILL . . . . .	581
9.49.2.8	DRV_HAS_DRAW_RECT_FRAME . . . . .	581
9.49.2.9	DRV_HAS_DRAW_RECT_ROUND_FILL . . . . .	581
9.49.2.10	DRV_HAS_DRAW_RECT_ROUND_FRAME . . . . .	581
9.49.2.11	DRV_HAS_DRAW_TEXT . . . . .	581
9.49.2.12	DRV_HAS_DRAW_TRI_FILL . . . . .	581
9.49.2.13	DRV_HAS_DRAW_TRI_FRAME . . . . .	582
9.49.2.14	DRV_OVERRIDE_TXT_ALIGN . . . . .	582
9.49.3	Function Documentation . . . . .	582
9.49.3.1	gslc_DrvAdaptColorToRaw() . . . . .	582
9.49.3.2	gslc_DrvDestruct() . . . . .	582
9.49.3.3	gslc_DrvDrawBkgnd() . . . . .	583
9.49.3.4	gslc_DrvDrawBmp24FromMem() . . . . .	583
9.49.3.5	gslc_DrvDrawBmp24FromSD() . . . . .	584
9.49.3.6	gslc_DrvDrawFillCircle() . . . . .	584
9.49.3.7	gslc_DrvDrawFillRect() . . . . .	585
9.49.3.8	gslc_DrvDrawFillRoundRect() . . . . .	585
9.49.3.9	gslc_DrvDrawFillTriangle() . . . . .	585
9.49.3.10	gslc_DrvDrawFrameCircle() . . . . .	586
9.49.3.11	gslc_DrvDrawFrameRect() . . . . .	587
9.49.3.12	gslc_DrvDrawFrameRoundRect() . . . . .	587
9.49.3.13	gslc_DrvDrawFrameTriangle() . . . . .	587
9.49.3.14	gslc_DrvDrawImage() . . . . .	588
9.49.3.15	gslc_DrvDrawLine() . . . . .	589
9.49.3.16	gslc_DrvDrawMonoFromMem() . . . . .	589
9.49.3.17	gslc_DrvDrawPoint() . . . . .	590
9.49.3.18	gslc_DrvDrawPoints() . . . . .	590
9.49.3.19	gslc_DrvDrawTxt() . . . . .	591
9.49.3.20	gslc_DrvFontAdd() . . . . .	591
9.49.3.21	gslc_DrvFontsDestruct() . . . . .	592

9.49.3.22	<a href="#">gslc_DrvGetDriverDisp()</a>	592
9.49.3.23	<a href="#">gslc_DrvGetDriverTouch()</a>	592
9.49.3.24	<a href="#">gslc_DrvGetNameDisp()</a>	593
9.49.3.25	<a href="#">gslc_DrvGetNameTouch()</a>	593
9.49.3.26	<a href="#">gslc_DrvGetTouch()</a>	594
9.49.3.27	<a href="#">gslc_DrvGetTxtSize()</a>	594
9.49.3.28	<a href="#">gslc_DrvImageDestruct()</a>	595
9.49.3.29	<a href="#">gslc_DrvInit()</a>	595
9.49.3.30	<a href="#">gslc_DrvInitTouch()</a>	596
9.49.3.31	<a href="#">gslc_DrvInitTs()</a>	596
9.49.3.32	<a href="#">gslc_DrvLoadImage()</a>	596
9.49.3.33	<a href="#">gslc_DrvPageFlipNow()</a>	597
9.49.3.34	<a href="#">gslc_DrvRotate()</a>	597
9.49.3.35	<a href="#">gslc_DrvSetBkgndColor()</a>	598
9.49.3.36	<a href="#">gslc_DrvSetBkgndImage()</a>	598
9.49.3.37	<a href="#">gslc_DrvSetClipRect()</a>	598
9.49.3.38	<a href="#">gslc_DrvSetElemImageGlow()</a>	599
9.49.3.39	<a href="#">gslc_DrvSetElemImageNorm()</a>	599
9.50	<a href="#">src/GUIslice_drv_m5stack.cpp File Reference</a>	600
9.51	<a href="#">src/GUIslice_drv_m5stack.h File Reference</a>	600
9.51.1	<a href="#">Detailed Description</a>	603
9.51.2	<a href="#">Macro Definition Documentation</a>	603
9.51.2.1	<a href="#">DRV_HAS_DRAW_BMP_MEM</a>	603
9.51.2.2	<a href="#">DRV_HAS_DRAW_CIRCLE_FILL</a>	603
9.51.2.3	<a href="#">DRV_HAS_DRAW_CIRCLE_FRAME</a>	604
9.51.2.4	<a href="#">DRV_HAS_DRAW_LINE</a>	604
9.51.2.5	<a href="#">DRV_HAS_DRAW_POINT</a>	604
9.51.2.6	<a href="#">DRV_HAS_DRAW_POINTS</a>	604
9.51.2.7	<a href="#">DRV_HAS_DRAW_RECT_FILL</a>	604
9.51.2.8	<a href="#">DRV_HAS_DRAW_RECT_FRAME</a>	604

9.51.2.9	DRV_HAS_DRAW_RECT_ROUND_FILL . . . . .	605
9.51.2.10	DRV_HAS_DRAW_RECT_ROUND_FRAME . . . . .	605
9.51.2.11	DRV_HAS_DRAW_TEXT . . . . .	605
9.51.2.12	DRV_HAS_DRAW_TRI_FILL . . . . .	605
9.51.2.13	DRV_HAS_DRAW_TRI_FRAME . . . . .	605
9.51.2.14	DRV_OVERRIDE_TXT_ALIGN . . . . .	605
9.51.3	Function Documentation . . . . .	606
9.51.3.1	gslc_DrvAdaptColorToRaw() . . . . .	606
9.51.3.2	gslc_DrvDestruct() . . . . .	606
9.51.3.3	gslc_DrvDrawBkgnd() . . . . .	606
9.51.3.4	gslc_DrvDrawBmp24FromMem() . . . . .	607
9.51.3.5	gslc_DrvDrawFillCircle() . . . . .	607
9.51.3.6	gslc_DrvDrawFillRect() . . . . .	608
9.51.3.7	gslc_DrvDrawFillRoundRect() . . . . .	608
9.51.3.8	gslc_DrvDrawFillTriangle() . . . . .	609
9.51.3.9	gslc_DrvDrawFrameCircle() . . . . .	609
9.51.3.10	gslc_DrvDrawFrameRect() . . . . .	610
9.51.3.11	gslc_DrvDrawFrameRoundRect() . . . . .	610
9.51.3.12	gslc_DrvDrawFrameTriangle() . . . . .	611
9.51.3.13	gslc_DrvDrawImage() . . . . .	611
9.51.3.14	gslc_DrvDrawLine() . . . . .	612
9.51.3.15	gslc_DrvDrawMonoFromMem() . . . . .	612
9.51.3.16	gslc_DrvDrawPoint() . . . . .	613
9.51.3.17	gslc_DrvDrawPoints() . . . . .	613
9.51.3.18	gslc_DrvDrawTxt() . . . . .	614
9.51.3.19	gslc_DrvDrawTxtAlign() . . . . .	614
9.51.3.20	gslc_DrvFontAdd() . . . . .	615
9.51.3.21	gslc_DrvFontsDestruct() . . . . .	615
9.51.3.22	gslc_DrvGetDriverDisp() . . . . .	616
9.51.3.23	gslc_DrvGetDriverTouch() . . . . .	616

9.51.3.24	<a href="#">gslc_DrvGetNameDisp()</a>	617
9.51.3.25	<a href="#">gslc_DrvGetNameTouch()</a>	617
9.51.3.26	<a href="#">gslc_DrvGetTxtSize()</a>	617
9.51.3.27	<a href="#">gslc_DrvImageDestruct()</a>	618
9.51.3.28	<a href="#">gslc_DrvInit()</a>	618
9.51.3.29	<a href="#">gslc_DrvInitTs()</a>	619
9.51.3.30	<a href="#">gslc_DrvLoadImage()</a>	619
9.51.3.31	<a href="#">gslc_DrvPageFlipNow()</a>	620
9.51.3.32	<a href="#">gslc_DrvRotate()</a>	620
9.51.3.33	<a href="#">gslc_DrvSetBkgndColor()</a>	621
9.51.3.34	<a href="#">gslc_DrvSetBkgndImage()</a>	621
9.51.3.35	<a href="#">gslc_DrvSetClipRect()</a>	621
9.51.3.36	<a href="#">gslc_DrvSetElemImageGlow()</a>	622
9.51.3.37	<a href="#">gslc_DrvSetElemImageNorm()</a>	622
9.51.4	<a href="#">Variable Documentation</a>	623
9.51.4.1	<a href="#">ERRSTR_NULL</a>	623
9.51.4.2	<a href="#">ERRSTR_PXD_NULL</a>	623
9.52	<a href="#">src/GUISlice_drv_sdl.c File Reference</a>	623
9.53	<a href="#">src/GUISlice_drv_sdl.h File Reference</a>	623
9.53.1	<a href="#">Detailed Description</a>	626
9.53.2	<a href="#">Macro Definition Documentation</a>	626
9.53.2.1	<a href="#">DRV_HAS_DRAW_POINT</a>	626
9.53.2.2	<a href="#">DRV_OVERRIDE_TXT_ALIGN</a>	626
9.53.3	<a href="#">Function Documentation</a>	626
9.53.3.1	<a href="#">gslc_DrvAdaptColor()</a>	626
9.53.3.2	<a href="#">gslc_DrvAdaptRect()</a>	627
9.53.3.3	<a href="#">gslc_DrvCleanStart()</a>	627
9.53.3.4	<a href="#">gslc_DrvDestruct()</a>	627
9.53.3.5	<a href="#">gslc_DrvDrawBkgnd()</a>	628
9.53.3.6	<a href="#">gslc_DrvDrawFillRect()</a>	628

9.53.3.7	<a href="#">gslc_DrvDrawFrameRect()</a>	628
9.53.3.8	<a href="#">gslc_DrvDrawImage()</a>	630
9.53.3.9	<a href="#">gslc_DrvDrawLine()</a>	630
9.53.3.10	<a href="#">gslc_DrvDrawPoint()</a>	631
9.53.3.11	<a href="#">gslc_DrvDrawPoints()</a>	631
9.53.3.12	<a href="#">gslc_DrvDrawTxt()</a>	632
9.53.3.13	<a href="#">gslc_DrvFontAdd()</a>	632
9.53.3.14	<a href="#">gslc_DrvFontsDestruct()</a>	633
9.53.3.15	<a href="#">gslc_DrvGetDriverDisp()</a>	633
9.53.3.16	<a href="#">gslc_DrvGetDriverTouch()</a>	634
9.53.3.17	<a href="#">gslc_DrvGetNameDisp()</a>	634
9.53.3.18	<a href="#">gslc_DrvGetNameTouch()</a>	634
9.53.3.19	<a href="#">gslc_DrvGetTouch()</a>	635
9.53.3.20	<a href="#">gslc_DrvGetTxtSize()</a>	635
9.53.3.21	<a href="#">gslc_DrvImageDestruct()</a>	636
9.53.3.22	<a href="#">gslc_DrvInit()</a>	636
9.53.3.23	<a href="#">gslc_DrvInitTouch()</a>	637
9.53.3.24	<a href="#">gslc_DrvLoadImage()</a>	637
9.53.3.25	<a href="#">gslc_DrvPageFlipNow()</a>	638
9.53.3.26	<a href="#">gslc_DrvReportInfoPost()</a>	638
9.53.3.27	<a href="#">gslc_DrvReportInfoPre()</a>	638
9.53.3.28	<a href="#">gslc_DrvRotate()</a>	638
9.53.3.29	<a href="#">gslc_DrvSetBkgndColor()</a>	639
9.53.3.30	<a href="#">gslc_DrvSetBkgndImage()</a>	639
9.53.3.31	<a href="#">gslc_DrvSetClipRect()</a>	640
9.53.3.32	<a href="#">gslc_DrvSetElemImageGlow()</a>	640
9.53.3.33	<a href="#">gslc_DrvSetElemImageNorm()</a>	640
9.54	<a href="#">src/GUISlice_drv_tft_espi.cpp File Reference</a>	641
9.55	<a href="#">src/GUISlice_drv_tft_espi.h File Reference</a>	641
9.55.1	<a href="#">Detailed Description</a>	644

9.55.2	Macro Definition Documentation	644
9.55.2.1	DRV_HAS_DRAW_BMP_MEM	644
9.55.2.2	DRV_HAS_DRAW_CIRCLE_FILL	645
9.55.2.3	DRV_HAS_DRAW_CIRCLE_FRAME	645
9.55.2.4	DRV_HAS_DRAW_LINE	645
9.55.2.5	DRV_HAS_DRAW_POINT	645
9.55.2.6	DRV_HAS_DRAW_POINTS	645
9.55.2.7	DRV_HAS_DRAW_RECT_FILL	645
9.55.2.8	DRV_HAS_DRAW_RECT_FRAME	646
9.55.2.9	DRV_HAS_DRAW_RECT_ROUND_FILL	646
9.55.2.10	DRV_HAS_DRAW_RECT_ROUND_FRAME	646
9.55.2.11	DRV_HAS_DRAW_TEXT	646
9.55.2.12	DRV_HAS_DRAW_TRI_FILL	646
9.55.2.13	DRV_HAS_DRAW_TRI_FRAME	646
9.55.2.14	DRV_OVERRIDE_TXT_ALIGN	647
9.55.2.15	GSLC_SPIFFS_EN	647
9.55.3	Function Documentation	647
9.55.3.1	gslc_DrvAdaptColorToRaw()	647
9.55.3.2	gslc_DrvDestruct()	647
9.55.3.3	gslc_DrvDrawBkgnd()	647
9.55.3.4	gslc_DrvDrawBmp24FromMem()	649
9.55.3.5	gslc_DrvDrawFillCircle()	649
9.55.3.6	gslc_DrvDrawFillRect()	650
9.55.3.7	gslc_DrvDrawFillRoundRect()	650
9.55.3.8	gslc_DrvDrawFillTriangle()	651
9.55.3.9	gslc_DrvDrawFrameCircle()	651
9.55.3.10	gslc_DrvDrawFrameRect()	652
9.55.3.11	gslc_DrvDrawFrameRoundRect()	652
9.55.3.12	gslc_DrvDrawFrameTriangle()	653
9.55.3.13	gslc_DrvDrawImage()	653

9.55.3.14	<a href="#">gslc_DrvDrawLine()</a>	654
9.55.3.15	<a href="#">gslc_DrvDrawMonoFromMem()</a>	654
9.55.3.16	<a href="#">gslc_DrvDrawPoint()</a>	655
9.55.3.17	<a href="#">gslc_DrvDrawPoints()</a>	655
9.55.3.18	<a href="#">gslc_DrvDrawTxt()</a>	656
9.55.3.19	<a href="#">gslc_DrvDrawTxtAlign()</a>	656
9.55.3.20	<a href="#">gslc_DrvFontAdd()</a>	657
9.55.3.21	<a href="#">gslc_DrvFontsDestruct()</a>	657
9.55.3.22	<a href="#">gslc_DrvGetDriverDisp()</a>	658
9.55.3.23	<a href="#">gslc_DrvGetDriverTouch()</a>	658
9.55.3.24	<a href="#">gslc_DrvGetNameDisp()</a>	659
9.55.3.25	<a href="#">gslc_DrvGetNameTouch()</a>	659
9.55.3.26	<a href="#">gslc_DrvGetTxtSize()</a>	659
9.55.3.27	<a href="#">gslc_DrvImageDestruct()</a>	660
9.55.3.28	<a href="#">gslc_DrvInit()</a>	660
9.55.3.29	<a href="#">gslc_DrvInitTs()</a>	661
9.55.3.30	<a href="#">gslc_DrvLoadImage()</a>	661
9.55.3.31	<a href="#">gslc_DrvPageFlipNow()</a>	662
9.55.3.32	<a href="#">gslc_DrvRotate()</a>	662
9.55.3.33	<a href="#">gslc_DrvSetBkgndColor()</a>	663
9.55.3.34	<a href="#">gslc_DrvSetBkgndImage()</a>	663
9.55.3.35	<a href="#">gslc_DrvSetClipRect()</a>	663
9.55.3.36	<a href="#">gslc_DrvSetElemImageGlow()</a>	664
9.55.3.37	<a href="#">gslc_DrvSetElemImageNorm()</a>	664
9.56	<a href="#">src/GUISlice_drv_utft.cpp File Reference</a>	665
9.57	<a href="#">src/GUISlice_drv_utft.h File Reference</a>	665
9.57.1	<a href="#">Detailed Description</a>	668
9.57.2	<a href="#">Macro Definition Documentation</a>	668
9.57.2.1	<a href="#">DRV_HAS_DRAW_BMP_MEM</a>	668
9.57.2.2	<a href="#">DRV_HAS_DRAW_CIRCLE_FILL</a>	668



9.57.2.3	DRV_HAS_DRAW_CIRCLE_FRAME . . . . .	668
9.57.2.4	DRV_HAS_DRAW_LINE . . . . .	669
9.57.2.5	DRV_HAS_DRAW_POINT . . . . .	669
9.57.2.6	DRV_HAS_DRAW_POINTS . . . . .	669
9.57.2.7	DRV_HAS_DRAW_RECT_FILL . . . . .	669
9.57.2.8	DRV_HAS_DRAW_RECT_FRAME . . . . .	669
9.57.2.9	DRV_HAS_DRAW_RECT_ROUND_FILL . . . . .	669
9.57.2.10	DRV_HAS_DRAW_RECT_ROUND_FRAME . . . . .	670
9.57.2.11	DRV_HAS_DRAW_TEXT . . . . .	670
9.57.2.12	DRV_HAS_DRAW_TRI_FILL . . . . .	670
9.57.2.13	DRV_HAS_DRAW_TRI_FRAME . . . . .	670
9.57.2.14	DRV_OVERRIDE_TXT_ALIGN . . . . .	670
9.57.3	Function Documentation . . . . .	670
9.57.3.1	gslc_DrvAdaptColorToRaw() . . . . .	670
9.57.3.2	gslc_DrvDestruct() . . . . .	670
9.57.3.3	gslc_DrvDrawBkgnd() . . . . .	671
9.57.3.4	gslc_DrvDrawBmp24FromMem() . . . . .	671
9.57.3.5	gslc_DrvDrawFillCircle() . . . . .	672
9.57.3.6	gslc_DrvDrawFillRect() . . . . .	672
9.57.3.7	gslc_DrvDrawFillRoundRect() . . . . .	673
9.57.3.8	gslc_DrvDrawFillTriangle() . . . . .	673
9.57.3.9	gslc_DrvDrawFrameCircle() . . . . .	674
9.57.3.10	gslc_DrvDrawFrameRect() . . . . .	674
9.57.3.11	gslc_DrvDrawFrameRoundRect() . . . . .	675
9.57.3.12	gslc_DrvDrawFrameTriangle() . . . . .	675
9.57.3.13	gslc_DrvDrawImage() . . . . .	676
9.57.3.14	gslc_DrvDrawLine() . . . . .	676
9.57.3.15	gslc_DrvDrawMonoFromMem() . . . . .	677
9.57.3.16	gslc_DrvDrawPoint() . . . . .	677
9.57.3.17	gslc_DrvDrawPoints() . . . . .	678

9.57.3.18	<a href="#">gslc_DrvDrawTxt()</a>	678
9.57.3.19	<a href="#">gslc_DrvFontAdd()</a>	679
9.57.3.20	<a href="#">gslc_DrvFontsDestruct()</a>	679
9.57.3.21	<a href="#">gslc_DrvGetDriverDisp()</a>	679
9.57.3.22	<a href="#">gslc_DrvGetDriverTouch()</a>	680
9.57.3.23	<a href="#">gslc_DrvGetNameDisp()</a>	680
9.57.3.24	<a href="#">gslc_DrvGetNameTouch()</a>	681
9.57.3.25	<a href="#">gslc_DrvGetTouch()</a>	681
9.57.3.26	<a href="#">gslc_DrvGetTxtSize()</a>	682
9.57.3.27	<a href="#">gslc_DrvImageDestruct()</a>	682
9.57.3.28	<a href="#">gslc_DrvInit()</a>	683
9.57.3.29	<a href="#">gslc_DrvInitTouch()</a>	683
9.57.3.30	<a href="#">gslc_DrvInitTs()</a>	684
9.57.3.31	<a href="#">gslc_DrvLoadImage()</a>	684
9.57.3.32	<a href="#">gslc_DrvPageFlipNow()</a>	684
9.57.3.33	<a href="#">gslc_DrvRotate()</a>	685
9.57.3.34	<a href="#">gslc_DrvSetBkgndColor()</a>	685
9.57.3.35	<a href="#">gslc_DrvSetBkgndImage()</a>	686
9.57.3.36	<a href="#">gslc_DrvSetClipRect()</a>	686
9.57.3.37	<a href="#">gslc_DrvSetElemImageGlow()</a>	687
9.57.3.38	<a href="#">gslc_DrvSetElemImageNorm()</a>	687
9.58	<a href="#">src/GUISlice_ex.h File Reference</a>	687
9.59	<a href="#">src/GUISlice_th.cpp File Reference</a>	688
9.59.1	<a href="#">Function Documentation</a>	688
9.59.1.1	<a href="#">gslc_getTouchHandler()</a>	689
9.59.1.2	<a href="#">gslc_InitTouchHandler()</a>	689
9.59.2	<a href="#">Variable Documentation</a>	689
9.59.2.1	<a href="#">pTouchHandler</a>	689
9.60	<a href="#">src/GUISlice_th.h File Reference</a>	689
9.60.1	<a href="#">Function Documentation</a>	690
9.60.1.1	<a href="#">gslc_getTouchHandler()</a>	690
9.60.1.2	<a href="#">gslc_InitTouchHandler()</a>	690
9.61	<a href="#">src/GUISlice_th_XPT2046.h File Reference</a>	690
9.62	<a href="#">src/GUISlice_version.h File Reference</a>	691
9.62.1	<a href="#">Macro Definition Documentation</a>	691
9.62.1.1	<a href="#">GUISLICE_VER</a>	691

# Chapter 1

## GUIslice library

*A lightweight GUI framework for embedded displays*

Design your GUI with a **drag & drop builder**, then apply the same code to a wide range of displays, libraries and controllers with the **cross-platform framework**. Open source **MIT license** grants free commercial usage.

- Extensive [Documentation](#) guides available
- [GUIslice API documentation \(online\) & \(PDF\)](#)
- Active development: see [latest updates & work in progress](#)
- [Release history](#)
- [Website \(www.impulseadventure.com\)](#)
- **Support email:** [guislice@gmail.com](mailto:guislice@gmail.com)
- GUIslice by Calvin Hass and [GitHub contributors](#), Builder by Paul Conti

### Features

- Pure C library, no dynamic memory allocation
- *Widgets:*
  - text, images, buttons, checkboxes, radio buttons, sliders, custom keypads, listbox, radial controls, scrolling textbox / terminal, graphs, etc. plus extensions and multiple pages.
- Cross-platform **GUIslice Builder** application to generate layouts
- *Platform-independent* GUI core currently supports:
  - Adafruit-GFX, TFT\_eSPI, mcufriend, UTFT, LCDGFX, SDL1.2, SDL2.0
- *Devices:*
  - Raspberry Pi, Arduino, ATmega2560, ESP8266 / NodeMCU, ESP32, M5stack, Teensy 3 / T4, WIO Terminal, Feather M0 (Cortex-M0), nRF52 (Cortex-M4F), LINUX, Beaglebone Black, STM32, Due, etc.
- *Typical displays:*

- PiTFT, Adafruit TFT 3.5" / 2.8" / 2.4" / 2.2" / 1.44", FeatherWing TFT, OLED 0.96", mcufriend, BuyDisplay / EastRising 4.3" 5" 7", Waveshare, 4D Cape
- *Display drivers include:*
  - ILI9341, ST7735, SSD1306, HX8347D, HX8357, PCD8544, RA8875, RA8876, ILI9225, ILI9341\_t3, ILI9341\_due
- *Touchscreen control including:*
  - STMPE610, FT6206, FT5206, XPT2046, 4-wire, tslib, URTouch, Adafruit Seesaw
- *IDE Support:*
  - GUIslice has been tested for use in the *Arduino IDE* and *Platform IO* environments, in addition to LINUX make
- Foreign characters / UTF-8 encoding (in SDL mode), anti-aliased fonts (in TFT\_eSPI mode)
- Dynamic display rotation
- GPIO / pin / keyboard / Adafruit Seesaw navigation for non-touchscreen devices

## Screenshots

## GUIslice Builder

- Includes cross-platform (Windows, LINUX and Mac) desktop application to generate GUIslice layouts
- Please refer to [GUIslice Builder wiki](#) for documentation

## Disclaimer

The Software is not designed for use in devices or situations where there may be physical injury if the Software has errors.

## Chapter 2

### Todo List

Global `gslc_ElemGetEditEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)

Doc

Global `gslc_ElemGetFocusEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)

Doc

Global `gslc_ElemSetFocusEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, bool bFocusEn)

Doc

Global `gslc_InputMapLookup` (`gslc_tsGui` \*pGui, `gslc_teInputRawEvent` eInputEvent, int16\_t nInputVal, `gslc_teAction` \*peAction, int16\_t \*pnActionVal)

Doc.

Global `gslc_XKeyPadFocusSetDefault` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)

Doc

Global `gslc_XKeyPadPendRedrawAddKey` (`gslc_tsXKeyPadResult` \*pResult, int16\_t nId)

Doc

Global `gslc_XKeyPadPendRedrawAddTxt` (`gslc_tsXKeyPadResult` \*pResult)

Doc

Global `gslc_XKeyPadPendRedrawReset` (`gslc_tsXKeyPadResult` \*pResult)

Doc

Global `gslc_XKeyPadRedrawUpdate` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)

Doc

Global `gslc_XKeyPadTrackSet` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, int16\_t nInd, `gslc_tsXKeyPadAttrib` eAttrib)

Doc



## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

General Functions . . . . .	13
Graphics General Functions . . . . .	21
Graphics Primitive Functions . . . . .	32
Font Functions . . . . .	43
Page Functions . . . . .	46
Element Functions . . . . .	53
Element: Creation Functions . . . . .	54
Element: General Functions . . . . .	60
Element: Update Functions . . . . .	61
Touchscreen Functions . . . . .	84
Input Mapping Functions . . . . .	90
General Purpose Macros . . . . .	95
Flash-based Element Macros . . . . .	97
Internal Functions . . . . .	103
Internal: Misc Functions . . . . .	133
Internal: Element Functions . . . . .	134
Internal: Page Functions . . . . .	142
Internal: Element Collection Functions . . . . .	148
Internal: Element Collection Event Functions . . . . .	154
Internal: Tracking Functions . . . . .	157
Internal: Cleanup Functions . . . . .	160





## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

gslc_tsCollect . . . . .	165
gslc_tsColor . . . . .	166
gslc_tsDriver . . . . .	166
gslc_tsElem . . . . .	168
gslc_tsElemRef . . . . .	170
gslc_tsEvent . . . . .	170
gslc_tsEventTouch . . . . .	171
gslc_tsFont . . . . .	171
gslc_tsGui . . . . .	172
gslc_tsImgRef . . . . .	175
gslc_tsInputMap . . . . .	176
gslc_tsKey . . . . .	176
gslc_tsLabelSpecial . . . . .	178
gslc_tsPage . . . . .	179
gslc_tsPt . . . . .	180
gslc_tsRect . . . . .	180
gslc_tsRectState . . . . .	181
gslc_tsXCheckbox . . . . .	182
gslc_tsXGauge . . . . .	183
gslc_tsXGlowball . . . . .	187
gslc_tsXGlowballRing . . . . .	190
gslc_tsXGraph . . . . .	191
gslc_tsXKeyPad . . . . .	195
gslc_tsXKeyPadCfg . . . . .	198
gslc_tsXKeyPadCfg_Alpha . . . . .	203
gslc_tsXKeyPadCfg_Num . . . . .	204
gslc_tsXKeyPadData . . . . .	205
gslc_tsXKeyPadResult . . . . .	206
gslc_tsXListbox . . . . .	208
gslc_tsXProgress . . . . .	213
gslc_tsXRadial . . . . .	215
gslc_tsXRamp . . . . .	219
gslc_tsXRingGauge . . . . .	220
gslc_tsXSeekBar . . . . .	223
gslc_tsXSelNum . . . . .	228

gslc_tsXSlider . . . . .	229
gslc_tsXSpinner . . . . .	232
gslc_tsXTemplate . . . . .	236
gslc_tsXTextbox . . . . .	236
gslc_tsXTogglebtn . . . . .	240
gslc_tsXToggleImgbtn . . . . .	242
THPoint . . . . .	243
TouchHandler . . . . .	245
TouchHandler_XPT2046 . . . . .	247

## Chapter 5

# Data Structure Index

### 5.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">gslc_tsCollect</a>	Element collection struct . . . . .	165
<a href="#">gslc_tsColor</a>	Color structure. Defines RGB triplet . . . . .	166
<a href="#">gslc_tsDriver</a>	. . . . .	166
<a href="#">gslc_tsElem</a>	Element Struct . . . . .	168
<a href="#">gslc_tsElemRef</a>	Element reference structure . . . . .	170
<a href="#">gslc_tsEvent</a>	Event structure . . . . .	170
<a href="#">gslc_tsEventTouch</a>	Structure used to pass touch data through event . . . . .	171
<a href="#">gslc_tsFont</a>	Font reference structure . . . . .	171
<a href="#">gslc_tsGui</a>	GUI structure . . . . .	172
<a href="#">gslc_tsImgRef</a>	Image reference structure . . . . .	175
<a href="#">gslc_tsInputMap</a>	Input mapping . . . . .	176
<a href="#">gslc_tsKey</a>	Key information. Defines everything we need to know about a particular key . . . . .	176
<a href="#">gslc_tsLabelSpecial</a>	Key Label strings for special buttons . . . . .	178
<a href="#">gslc_tsPage</a>	Page structure . . . . .	179
<a href="#">gslc_tsPt</a>	Define point coordinates . . . . .	180
<a href="#">gslc_tsRect</a>	Rectangular region. Defines X,Y corner coordinates plus dimensions . . . . .	180
<a href="#">gslc_tsRectState</a>	State associated with an element's region . . . . .	181
<a href="#">gslc_tsXCheckbox</a>	Extended data for Checkbox element . . . . .	182

<a href="#">gslc_tsXGauge</a>	
Extended data for Gauge element . . . . .	183
<a href="#">gslc_tsXGlowball</a>	
Extended data for Slider element . . . . .	187
<a href="#">gslc_tsXGlowballRing</a> . . . . .	190
<a href="#">gslc_tsXGraph</a>	
Extended data for Graph element . . . . .	191
<a href="#">gslc_tsXKeyPad</a>	
Extended data for KeyPad element . . . . .	195
<a href="#">gslc_tsXKeyPadCfg</a>	
Configuration for the KeyPad . . . . .	198
<a href="#">gslc_tsXKeyPadCfg_Alpha</a> . . . . .	203
<a href="#">gslc_tsXKeyPadCfg_Num</a> . . . . .	204
<a href="#">gslc_tsXKeyPadData</a>	
Input callback data structure . . . . .	205
<a href="#">gslc_tsXKeyPadResult</a>	
Return status for XKeyPad . . . . .	206
<a href="#">gslc_tsXListbox</a>	
Extended data for Listbox element . . . . .	208
<a href="#">gslc_tsXProgress</a>	
Extended data for Gauge element . . . . .	213
<a href="#">gslc_tsXRadial</a>	
Extended data for Gauge element . . . . .	215
<a href="#">gslc_tsXRamp</a>	
Extended data for Gauge element . . . . .	219
<a href="#">gslc_tsXRingGauge</a>	
Extended data for XRingGauge element . . . . .	220
<a href="#">gslc_tsXSeekbar</a>	
Extended data for Seekbar element . . . . .	223
<a href="#">gslc_tsXSelNum</a>	
Extended data for SelNum element . . . . .	228
<a href="#">gslc_tsXSlider</a>	
Extended data for Slider element . . . . .	229
<a href="#">gslc_tsXSpinner</a>	
Extended data for Spinner element . . . . .	232
<a href="#">gslc_tsXTemplate</a>	
Callback function for slider feedback . . . . .	236
<a href="#">gslc_tsXTextbox</a>	
Extended data for Textbox element . . . . .	236
<a href="#">gslc_tsXTogglebtn</a>	
Extended data for Togglebtn element . . . . .	240
<a href="#">gslc_tsXToggleImgbtn</a>	
Extended data for ToggleImgbtn element . . . . .	242
<a href="#">THPoint</a> . . . . .	243
<a href="#">TouchHandler</a> . . . . .	245
<a href="#">TouchHandler_XPT2046</a> . . . . .	247

## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

src/GUISlice.c	524
src/GUISlice.h	536
src/GUISlice_config.h	576
src/GUISlice_drv.h	576
src/GUISlice_drv_adagfx.cpp	576
src/GUISlice_drv_adagfx.h	576
GUISlice library (driver layer for Adafruit-GFX)	577
src/GUISlice_drv_m5stack.cpp	600
src/GUISlice_drv_m5stack.h	600
GUISlice library (driver layer for M5stack)	600
src/GUISlice_drv_sdl.c	623
src/GUISlice_drv_sdl.h	623
GUISlice library (driver layer for LINUX / SDL)	623
src/GUISlice_drv_tft_espi.cpp	641
src/GUISlice_drv_tft_espi.h	641
GUISlice library (driver layer for TFT-eSPI)	641
src/GUISlice_drv_utft.cpp	665
src/GUISlice_drv_utft.h	665
GUISlice library (driver layer for UTFT)	665
src/GUISlice_ex.h	687
src/GUISlice_th.cpp	688
src/GUISlice_th.h	689
src/GUISlice_th_XPT2046.h	690
src/GUISlice_version.h	691
src/elem/XCheckbox.c	251
src/elem/XCheckbox.h	256
src/elem/XGauge.c	263
src/elem/XGauge.h	269
src/elem/XGlowball.c	276
src/elem/XGlowball.h	281
src/elem/XGraph.c	285
src/elem/XGraph.h	289
src/elem/XKeyPad.c	295
src/elem/XKeyPad.h	313
src/elem/XKeyPad_Alpha-setup.h	336

src/elem/XKeyPad_Alpha.c	343
src/elem/XKeyPad_Alpha.h	348
src/elem/XKeyPad_Num-setup.h	353
src/elem/XKeyPad_Num.c	361
src/elem/XKeyPad_Num.h	367
src/elem/XListbox.c	373
src/elem/XListbox.h	383
src/elem/XProgress.c	394
src/elem/XProgress.h	398
src/elem/XRadial.c	404
src/elem/XRadial.h	409
src/elem/XRamp.c	415
src/elem/XRamp.h	419
src/elem/XRingGauge.c	423
src/elem/XRingGauge.h	431
src/elem/XSeekbar.c	438
src/elem/XSeekbar.h	443
src/elem/XSelNum.c	450
src/elem/XSelNum.h	455
src/elem/XSlider.c	459
src/elem/XSlider.h	465
src/elem/XSpinner.c	
Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound	472
src/elem/XSpinner.h	
Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound	478
src/elem/XTemplate.c	484
src/elem/XTemplate.h	487
src/elem/XTextbox.c	489
src/elem/XTextbox.h	495
src/elem/XTogglebtn.c	501
src/elem/XTogglebtn.h	507
src/elem/XToggleImgbtn.c	514
src/elem/XToggleImgbtn.h	519

## Chapter 7

# Module Documentation

### 7.1 General Functions

General functions for configuring the GUI.

#### Functions

- `char * gslc_GetVer (gslc_tsGui *pGui)`  
*Get the GUIslice version number.*
- `const char * gslc_GetNameDisp (gslc_tsGui *pGui)`  
*Get the GUIslice display driver name.*
- `const char * gslc_GetNameTouch (gslc_tsGui *pGui)`  
*Get the GUIslice touch driver name.*
- `void * gslc_GetDriverDisp (gslc_tsGui *pGui)`  
*Get the native display driver instance.*
- `void * gslc_GetDriverTouch (gslc_tsGui *pGui)`  
*Get the native touch driver instance.*
- `bool gslc_Init (gslc_tsGui *pGui, void *pvDriver, gslc_tsPage *asPage, uint8_t nMaxPage, gslc_tsFont *asFont, uint8_t nMaxFont)`  
*Initialize the GUIslice library.*
- `void gslc_InitDebug (GSLC_CB_DEBUG_OUT pfunc)`  
*Initialize debug output.*
- `void gslc_DebugPrintf (const char *pFmt,...)`  
*Optimized printf routine for GUIslice debug/error output.*
- `bool gslc_GuiRotate (gslc_tsGui *pGui, uint8_t nRotation)`  
*Dynamically change rotation, automatically adapt touchscreen axes swap/flip.*
- `void gslc_Quit (gslc_tsGui *pGui)`  
*Exit the GUIslice environment.*
- `void gslc_Update (gslc_tsGui *pGui)`  
*Perform main GUIslice handling functions.*
- `bool gslc_SetBkgndImage (gslc_tsGui *pGui, gslc_tsImgRef sImgRef)`  
*Configure the background to use a bitmap image.*
- `bool gslc_SetBkgndColor (gslc_tsGui *pGui, gslc_tsColor nCol)`  
*Configure the background to use a solid color.*
- `bool gslc_SetTransparentColor (gslc_tsGui *pGui, gslc_tsColor nCol)`  
*Configure the color to use for image transparency.*
- `gslc_tsRect gslc_GetClipRect (gslc_tsGui *pGui)`  
*Get the current the clipping rectangle.*
- `bool gslc_SetClipRect (gslc_tsGui *pGui, gslc_tsRect *pRect)`  
*Set the clipping rectangle for further drawing.*

### 7.1.1 Detailed Description

General functions for configuring the GUI.

### 7.1.2 Function Documentation

#### 7.1.2.1 `gslc_DebugPrintf()`

```
void gslc_DebugPrintf (
    const char * pFmt,
    ... )
```

Optimized printf routine for GUIslice debug/error output.

- Only supports 's','d','u' tokens
- Calls on the output function configured in [gslc\\_InitDebug\(\)](#)

##### Parameters

in	<i>pFmt</i>	Format string to use for printing
in	...	Variable parameter list

##### Returns

none

#### 7.1.2.2 `gslc_GetClipRect()`

```
gslc_tsRect gslc_GetClipRect (
    gslc_tsGui * pGui )
```

Get the current the clipping rectangle.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

##### Returns

rect for active clipping region



### 7.1.2.3 gslc\_GetDriverDisp()

```
void* gslc_GetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

### 7.1.2.4 gslc\_GetDriverTouch()

```
void* gslc_GetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

### 7.1.2.5 gslc\_GetNameDisp()

```
const char* gslc_GetNameDisp (
    gslc_tsGui * pGui )
```

Get the GUIslice display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**7.1.2.6 gslc\_GetNameTouch()**

```
const char* gslc_GetNameTouch (
    gslc_tsGui * pGui )
```

Get the GUIslice touch driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**7.1.2.7 gslc\_GetVer()**

```
char* gslc_GetVer (
    gslc_tsGui * pGui )
```

Get the GUIslice version number.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing version number

**7.1.2.8 gslc\_GuiRotate()**

```
bool gslc_GuiRotate (
    gslc_tsGui * pGui,
    uint8_t nRotation )
```

Dynamically change rotation, automatically adapt touchscreen axes swap/flip.

The function assumes that the touchscreen settings for swap and flip in the GUIslice config are valid for the configured GSLC\_ROTATE.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

#### Returns

true if success, false otherwise

#### 7.1.2.9 gslc\_Init()

```
bool gslc_Init (
    gslc_tsGui * pGui,
    void * pvDriver,
    gslc_tsPage * asPage,
    uint8_t nMaxPage,
    gslc_tsFont * asFont,
    uint8_t nMaxFont )
```

Initialize the GUIslice library.

- Configures the primary screen surface(s)
- Initializes font support

#### PRE:

- The environment variables should be configured before calling [gslc\\_Init\(\)](#).

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pvDriver</i>	Void pointer to Driver struct (gslc_tsDriver*)
in	<i>asPage</i>	Pointer to Page array
in	<i>nMaxPage</i>	Size of Page array
in	<i>asFont</i>	Pointer to Font array
in	<i>nMaxFont</i>	Size of Font array

#### Returns

true if success, false if fail

### 7.1.2.10 gslc\_InitDebug()

```
void gslc_InitDebug (
    GSLC_CB_DEBUG_OUT pfunc )
```

Initialize debug output.

- Defines the user function used for debug/error output
- pfunc is responsible for outputting a single character
- For Arduino, this user function would typically call Serial.print()

#### Parameters

in	<i>pfunc</i>	Pointer to user character-out function
----	--------------	--

#### Returns

none

### 7.1.2.11 gslc\_Quit()

```
void gslc_Quit (
    gslc_tsGui * pGui )
```

Exit the GUIslice environment.

- Calls lower-level destructors to clean up any initialized subsystems and deletes any created elements or fonts

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

None

### 7.1.2.12 gslc\_SetBkgndColor()

```
bool gslc_SetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

**Returns**

true if success, false if fail

**7.1.2.13 gslc\_SetBkgndImage()**

```
bool gslc_SetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if fail

**7.1.2.14 gslc\_SetClipRect()**

```
bool gslc_SetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for further drawing.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Pointer to Rect for clipping (or NULL for entire screen)

**Returns**

true if success, false if error

**7.1.2.15 gslc\_SetTransparentColor()**

```
bool gslc_SetTransparentColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the color to use for image transparency.

- Drawing a BMP with transparency enabled will cause regions in this specific color to appear transparent
- This API overrides the config option GSLC\_BMP\_TRANS\_RGB

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

**Returns**

true if success, false if fail

**7.1.2.16 gslc\_Update()**

```
void gslc_Update (
    gslc_tsGui * pGui )
```

Perform main GUIslice handling functions.

- Handles any touch events
- Performs any necessary screen redraw

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

None

## 7.2 Graphics General Functions

Helper functions that support graphics operations.

### Functions

- `bool gslc_IsInRect (int16_t nSelX, int16_t nSelY, gslc_tsRect rRect)`  
*Determine if a coordinate is inside of a rectangular region.*
- `gslc_tsRect gslc_ExpandRect (gslc_tsRect rRect, int16_t nExpandW, int16_t nExpandH)`  
*Expand or contract a rectangle in width and/or height (equal amounts on both side), based on the centerpoint of the rectangle.*
- `bool gslc_IsInWH (int16_t nSelX, int16_t nSelY, uint16_t nWidth, uint16_t nHeight)`  
*Determine if a coordinate is inside of a width x height region.*
- `void gslc_UnionRect (gslc_tsRect *pRect, gslc_tsRect rAddRect)`  
*Expand a rect to include another rect.*
- `void gslc_InvalidateRgnReset (gslc_tsGui *pGui)`  
*Reset the invalidation region.*
- `void gslc_InvalidateRgnPage (gslc_tsGui *pGui, gslc_tsPage *pPage)`  
*Include an entire page (eg.*
- `void gslc_InvalidateRgnScreen (gslc_tsGui *pGui)`  
*Mark the entire screen as invalidated.*
- `void gslc_InvalidateRgnAdd (gslc_tsGui *pGui, gslc_tsRect rAddRect)`  
*Add a rectangular region to the invalidation region.*
- `bool gslc_ClipPt (gslc_tsRect *pClipRect, int16_t nX, int16_t nY)`  
*Perform basic clipping of a single point to a clipping region.*
- `bool gslc_ClipLine (gslc_tsRect *pClipRect, int16_t *pnX0, int16_t *pnY0, int16_t *pnX1, int16_t *pnY1)`  
*Perform basic clipping of a line to a clipping region.*
- `bool gslc_ClipRect (gslc_tsRect *pClipRect, gslc_tsRect *pRect)`  
*Perform basic clipping of a rectangle to a clipping region.*
- `gslc_tslmgRef gslc_GetImageFromFile (const char *pFname, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap file in LINUX filesystem.*
- `gslc_tslmgRef gslc_GetImageFromSD (const char *pFname, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap file in SD card.*
- `gslc_tslmgRef gslc_GetImageFromRam (unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap in SRAM.*
- `gslc_tslmgRef gslc_GetImageFromProg (const unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap in program memory (PROGMEM)*
- `void gslc_PolarToXY (uint16_t nRad, int16_t n64Ang, int16_t *nDX, int16_t *nDY)`  
*Convert polar coordinate to cartesian.*
- `int16_t gslc_sinFX (int16_t n64Ang)`  
*Calculate fixed-point sine function from fractional degrees.*
- `int16_t gslc_cosFX (int16_t n64Ang)`  
*Calculate fixed-point cosine function from fractional degrees.*
- `gslc_tsColor gslc_ColorBlend2 (gslc_tsColor colStart, gslc_tsColor colEnd, uint16_t nMidAmt, uint16_t n← BlendAmt)`  
*Create a color based on a blend between two colors.*
- `gslc_tsColor gslc_ColorBlend3 (gslc_tsColor colStart, gslc_tsColor colMid, gslc_tsColor colEnd, uint16_t n← MidAmt, uint16_t nBlendAmt)`  
*Create a color based on a blend between three colors.*
- `bool gslc_ColorEqual (gslc_tsColor a, gslc_tsColor b)`  
*Check whether two colors are equal.*

## 7.2.1 Detailed Description

Helper functions that support graphics operations.

## 7.2.2 Function Documentation

### 7.2.2.1 `gslc_ClipLine()`

```
bool gslc_ClipLine (
    gslc_tsRect * pClipRect,
    int16_t * pnX0,
    int16_t * pnY0,
    int16_t * pnX1,
    int16_t * pnY1 )
```

Perform basic clipping of a line to a clipping region.

- Implements Cohen-Sutherland algorithm
- Coordinates in parameter list are modified to fit the region

#### Parameters

in	<i>pClipRect</i>	Pointer to clipping region
in, out	<i>pnX0</i>	Ptr to X coordinate of line start
in, out	<i>pnY0</i>	Ptr to Y coordinate of line start
in, out	<i>pnX1</i>	Ptr to X coordinate of line end
in, out	<i>pnY1</i>	Ptr to Y coordinate of line end

#### Returns

true if line is visible, false if it should be discarded

### 7.2.2.2 `gslc_ClipPt()`

```
bool gslc_ClipPt (
    gslc_tsRect * pClipRect,
    int16_t nX,
    int16_t nY )
```

Perform basic clipping of a single point to a clipping region.



**Parameters**

in	<i>pClipRect</i>	Pointer to clipping region
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point

**Returns**

true if point is visible, false if it should be discarded

**7.2.2.3 gslc\_ClipRect()**

```
bool gslc_ClipRect (
    gslc_tsRect * pClipRect,
    gslc_tsRect * pRect )
```

Perform basic clipping of a rectangle to a clipping region.

- Coordinates in parameter rect are modified to fit the region

**Parameters**

in	<i>pClipRect</i>	Pointer to clipping region
in, out	<i>pRect</i>	Ptr to rectangle

**Returns**

true if rect is visible, false if it should be discarded

**7.2.2.4 gslc\_ColorBlend2()**

```
gslc_tsColor gslc_ColorBlend2 (
    gslc_tsColor colStart,
    gslc_tsColor colEnd,
    uint16_t nMidAmt,
    uint16_t nBlendAmt )
```

Create a color based on a blend between two colors.

**Parameters**

in	<i>colStart</i>	Starting color
in	<i>colEnd</i>	Ending color
in	<i>nMidAmt</i>	Position (0..1000) between start and end color at which the midpoint between colors should appear. Normally set to 500 (half-way).
in	<i>nBlendAmt</i>	The position (0..1000) between start and end at which we want to calculate the resulting blended color.

**Returns**

Blended color

**7.2.2.5 gslc\_ColorBlend3()**

```
gslc_tsColor gslc_ColorBlend3 (
    gslc_tsColor colStart,
    gslc_tsColor colMid,
    gslc_tsColor colEnd,
    uint16_t nMidAmt,
    uint16_t nBlendAmt )
```

Create a color based on a blend between three colors.

**Parameters**

in	<i>colStart</i>	Starting color
in	<i>colMid</i>	Intermediate color
in	<i>colEnd</i>	Ending color
in	<i>nMidAmt</i>	Position (0..1000) between start and end color at which the intermediate color should appear.
in	<i>nBlendAmt</i>	The position (0..1000) between start and end at which we want to calculate the resulting blended color.

**Returns**

Blended color

**7.2.2.6 gslc\_ColorEqual()**

```
bool gslc_ColorEqual (
    gslc_tsColor a,
    gslc_tsColor b )
```

Check whether two colors are equal.

**Parameters**

in	<i>a</i>	First color
in	<i>b</i>	Second color

**Returns**

True iff a and b are the same color.

7.2.2.7 `gslc_cosFX()`

```
int16_t gslc_cosFX (
    int16_t n64Ang )
```

Calculate fixed-point cosine function from fractional degrees.

- Depending on configuration, the result is derived from either floating point math library or fixed point lookup table.
- `gslc_cosFX(nAngDeg*64)/32768.0 = cos(nAngDeg*2pi/360)`

## Parameters

in	<i>n64Ang</i>	Angle (in units of 1/64 degrees)
----	---------------	----------------------------------

## Returns

Fixed-point cosine result. Signed 16-bit; divide by 32768 to get the actual value.

7.2.2.8 `gslc_ExpandRect()`

```
gslc_tsRect gslc_ExpandRect (
    gslc_tsRect rRect,
    int16_t nExpandW,
    int16_t nExpandH )
```

Expand or contract a rectangle in width and/or height (equal amounts on both side), based on the centerpoint of the rectangle.

## Parameters

in	<i>rRect</i>	Rectangular region before resizing
in	<i>nExpandW</i>	Number of pixels to expand the width (if positive) or contract the width (if negative)
in	<i>nExpandH</i>	Number of pixels to expand the height (if positive) or contract the height (if negative)

## Returns

`gslc_tsRect()` with resized dimensions

7.2.2.9 `gslc_GetImageFromFile()`

```
gslc_tsImgRef gslc_GetImageFromFile (
    const char * pFname,
    gslc_teImgRefFlags eFmt )
```

Create an image reference to a bitmap file in LINUX filesystem.

**Parameters**

in	<i>pFname</i>	Pointer to filename string of image in filesystem
in	<i>eFmt</i>	Image format

**Returns**

Loaded image reference

**7.2.2.10 gslc\_GetImageFromProg()**

```
gslc_tsImgRef gslc_GetImageFromProg (
    const unsigned char * pImgBuf,
    gslc_teImgRefFlags eFmt )
```

Create an image reference to a bitmap in program memory (PROGMEM)

**Parameters**

in	<i>pImgBuf</i>	Pointer to image buffer in memory
in	<i>eFmt</i>	Image format

**Returns**

Loaded image reference

**7.2.2.11 gslc\_GetImageFromRam()**

```
gslc_tsImgRef gslc_GetImageFromRam (
    unsigned char * pImgBuf,
    gslc_teImgRefFlags eFmt )
```

Create an image reference to a bitmap in SRAM.

**Parameters**

in	<i>pImgBuf</i>	Pointer to image buffer in memory
in	<i>eFmt</i>	Image format

**Returns**

Loaded image reference

7.2.2.12 `gslc_GetImageFromSD()`

```
gslc_tsImgRef gslc_GetImageFromSD (
    const char * pFname,
    gslc_teImgRefFlags eFmt )
```

Create an image reference to a bitmap file in SD card.

## Parameters

in	<i>pFname</i>	Pointer to filename string of image in SD card
in	<i>eFmt</i>	Image format

## Returns

Loaded image reference

7.2.2.13 `gslc_InvalidateRgnAdd()`

```
void gslc_InvalidateRgnAdd (
    gslc_tsGui * pGui,
    gslc_tsRect rAddRect )
```

Add a rectangular region to the invalidation region.

- This is usually called when an element has been modified

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rAddRect</i>	Rectangle to add to the invalidation region

## Returns

none

7.2.2.14 `gslc_InvalidateRgnPage()`

```
void gslc_InvalidateRgnPage (
    gslc_tsGui * pGui,
    gslc_tsPage * pPage )
```

Include an entire page (eg.  
from a page stack) in the invalidation region

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pPage</i>	Pointer to page

**Returns**

none

**7.2.2.15 gslc\_InvalidateRgnReset()**

```
void gslc_InvalidateRgnReset (
    gslc_tsGui * pGui )
```

Reset the invalidation region.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**7.2.2.16 gslc\_InvalidateRgnScreen()**

```
void gslc_InvalidateRgnScreen (
    gslc_tsGui * pGui )
```

Mark the entire screen as invalidated.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**7.2.2.17 gslc\_IsInRect()**

```
bool gslc_IsInRect (
    int16_t nSelX,
```

```
int16_t nSelY,
gslc_tsRect rRect )
```

Determine if a coordinate is inside of a rectangular region.

- This routine is useful in determining if a touch coordinate is inside of a button.

#### Parameters

in	<i>nSelX</i>	X coordinate to test
in	<i>nSelY</i>	X coordinate to test
in	<i>rRect</i>	Rectangular region to compare against

#### Returns

true if inside region, false otherwise

#### 7.2.2.18 gslc\_IsInWH()

```
bool gslc_IsInWH (
    int16_t nSelX,
    int16_t nSelY,
    uint16_t nWidth,
    uint16_t nHeight )
```

Determine if a coordinate is inside of a width x height region.

- This routine is useful in determining if a relative coordinate is within a given W x H dimension

#### Parameters

in	<i>nSelX</i>	X coordinate to test
in	<i>nSelY</i>	X coordinate to test
in	<i>nWidth</i>	Width to test against
in	<i>nHeight</i>	Height to test against

#### Returns

true if inside region, false otherwise

#### 7.2.2.19 gslc\_PolarToXY()

```
void gslc_PolarToXY (
    uint16_t nRad,
```

```

    int16_t  n64Ang,
    int16_t * nDX,
    int16_t * nDY )

```

Convert polar coordinate to cartesian.

#### Parameters

in	<i>nRad</i>	Radius of ray
in	<i>n64Ang</i>	Angle of ray (in units of 1/64 degrees, 0 is up)
out	<i>nDX</i>	X offset for ray end
out	<i>nDY</i>	Y offset for ray end

#### Returns

none

#### 7.2.2.20 gslc\_sinFX()

```

int16_t gslc_sinFX (
    int16_t  n64Ang )

```

Calculate fixed-point sine function from fractional degrees.

- Depending on configuration, the result is derived from either floating point math library or fixed point lookup table.
- $\text{gslc\_sinFX}(\text{nAngDeg} * 64) / 32768.0 = \sin(\text{nAngDeg} * 2\pi / 360)$

#### Parameters

in	<i>n64Ang</i>	Angle (in units of 1/64 degrees)
----	---------------	----------------------------------

#### Returns

Fixed-point sine result. Signed 16-bit; divide by 32768 to get the actual value.

#### 7.2.2.21 gslc\_UnionRect()

```

void gslc_UnionRect (
    gslc_tsRect * pRect,
    gslc_tsRect rAddRect )

```

Expand a rect to include another rect.

- This routine can be useful to modify an invalidation region to include another modified element



## Parameters

in	<i>pRect</i>	Initial rect region
in	<i>rAddRect</i>	Rectangle to add to the rect region

## Returns

none

## 7.3 Graphics Primitive Functions

These routines cause immediate drawing to occur on the primary screen.

### Functions

- void `gslc_DrawSetPixel` (`gslc_tsGui` \*pGui, int16\_t nX, int16\_t nY, `gslc_tsColor` nCol)  
*Set a pixel on the active screen to the given color with lock.*
- void `gslc_DrawLine` (`gslc_tsGui` \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, `gslc_tsColor` nCol)  
*Draw an arbitrary line using Bresenham's algorithm.*
- void `gslc_DrawLineH` (`gslc_tsGui` \*pGui, int16\_t nX, int16\_t nY, uint16\_t nW, `gslc_tsColor` nCol)  
*Draw a horizontal line.*
- void `gslc_DrawLineV` (`gslc_tsGui` \*pGui, int16\_t nX, int16\_t nY, uint16\_t nH, `gslc_tsColor` nCol)  
*Draw a vertical line.*
- void `gslc_DrawLinePolar` (`gslc_tsGui` \*pGui, int16\_t nX, int16\_t nY, uint16\_t nRadStart, uint16\_t nRadEnd, int16\_t n64Ang, `gslc_tsColor` nCol)  
*Draw a polar ray segment.*
- void `gslc_DrawFrameRect` (`gslc_tsGui` \*pGui, `gslc_tsRect` rRect, `gslc_tsColor` nCol)  
*Draw a framed rectangle.*
- void `gslc_DrawFrameRoundRect` (`gslc_tsGui` \*pGui, `gslc_tsRect` rRect, int16\_t nRadius, `gslc_tsColor` nCol)  
*Draw a framed rounded rectangle.*
- void `gslc_DrawFillRect` (`gslc_tsGui` \*pGui, `gslc_tsRect` rRect, `gslc_tsColor` nCol)  
*Draw a filled rectangle.*
- void `gslc_DrawFillRoundRect` (`gslc_tsGui` \*pGui, `gslc_tsRect` rRect, int16\_t nRadius, `gslc_tsColor` nCol)  
*Draw a filled rounded rectangle.*
- void `gslc_DrawFrameCircle` (`gslc_tsGui` \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, `gslc_tsColor` nCol)  
*Draw a framed circle.*
- void `gslc_DrawFillCircle` (`gslc_tsGui` \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, `gslc_tsColor` nCol)  
*Draw a filled circle.*
- void `gslc_DrawFrameTriangle` (`gslc_tsGui` \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, `gslc_tsColor` nCol)  
*Draw a framed triangle.*
- void `gslc_DrawFillTriangle` (`gslc_tsGui` \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, `gslc_tsColor` nCol)  
*Draw a filled triangle.*
- void `gslc_DrawFrameQuad` (`gslc_tsGui` \*pGui, `gslc_tsPt` \*psPt, `gslc_tsColor` nCol)  
*Draw a framed quadrilateral.*
- void `gslc_DrawFillQuad` (`gslc_tsGui` \*pGui, `gslc_tsPt` \*psPt, `gslc_tsColor` nCol)  
*Draw a filled quadrilateral.*
- void `gslc_DrawFillGradSector` (`gslc_tsGui` \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, `gslc_tsColor` cArcStart, `gslc_tsColor` cArcEnd, int16\_t nAngSecStart, int16\_t nAngSecEnd, int16\_t nAngGradStart, int16\_t nAngGradRange)  
*Draw a gradient filled sector of a circle with support for inner and outer radius.*
- void `gslc_DrawFillSector` (`gslc_tsGui` \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, `gslc_tsColor` cArc, int16\_t nAngSecStart, int16\_t nAngSecEnd)  
*Draw a flat filled sector of a circle with support for inner and outer radius.*

### 7.3.1 Detailed Description

These routines cause immediate drawing to occur on the primary screen.

### 7.3.2 Function Documentation

#### 7.3.2.1 `gslc_DrawFillCircle()`

```
void gslc_DrawFillCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled circle.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center X coordinate
in	<i>nMidY</i>	Center Y coordinate
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value for the fill

##### Returns

none

#### 7.3.2.2 `gslc_DrawFillGradSector()`

```
void gslc_DrawFillGradSector (
    gslc_tsGui * pGui,
    int16_t nQuality,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nRad1,
    int16_t nRad2,
    gslc_tsColor cArcStart,
    gslc_tsColor cArcEnd,
    int16_t nAngSecStart,
    int16_t nAngSecEnd,
    int16_t nAngGradStart,
    int16_t nAngGradRange )
```

Draw a gradient filled sector of a circle with support for inner and outer radius.

- Can be used to create a ring or pie chart
- Note that the gradient fill is defined by both the color stops (`cArcStart..cArcEnd`) as well as a gradient angular range (`nAngGradStart..nAngGradStart+nAngGradRange`). This gradient angular range can be different from the drawing angular range (`nAngSegStart..nAngSegEnd`) to enable more advanced control styling / updates.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nQuality</i>	Number of segments used to depict a full circle. The higher the value, the smoother the resulting arcs. A value of 72 provides 360/72=5 degrees per segment which is a reasonable compromise between smoothness and performance. Note that 360/nQuality should be an integer result, thus the allowable quality settings are: 360 (max quality), 180, 120, 90, 72, 60, 45, 40, 36 (low quality), etc.
in	<i>nMidX</i>	Midpoint X coordinate of circle
in	<i>nMidY</i>	Midpoint Y coordinate of circle
in	<i>nRad1</i>	Inner sector radius (0 for sector / pie, non-zero for ring)
in	<i>nRad2</i>	Outer sector radius. Delta from nRad1 defines ring thickness.
in	<i>cArcStart</i>	Start color for gradient fill (with angular range defined by nAngGradStart,nAngGradRange)
in	<i>cArcEnd</i>	End color for gradient fill
in	<i>nAngSecStart</i>	Angle of start of sector drawing (0 at top), measured in degrees.
in	<i>nAngSecEnd</i>	Angle of end of sector drawing (0 at top), measured in degrees.
in	<i>nAngGradStart</i>	For gradient fill, defines the starting angle associated with the starting color ( <i>cArcStart</i> )
in	<i>nAngGradRange</i>	For gradient fill, defines the angular range associated with the start-to-end color range ( <i>cArcStart..cArcEnd</i> )

#### Returns

none

#### 7.3.2.3 gslc\_DrawFillQuad()

```
void gslc_DrawFillQuad (
    gslc_tsGui * pGui,
    gslc_tsPt * psPt,
    gslc_tsColor nCol )
```

Draw a filled quadrilateral.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>psPt</i>	Pointer to array of 4 points
in	<i>nCol</i>	Color RGB value for the frame

**Returns**

true if success, false if error

**7.3.2.4 gslc\_DrawFillRect()**

```
void gslc_DrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

**Returns**

none

**7.3.2.5 gslc\_DrawFillRoundRect()**

```
void gslc_DrawFillRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nRadius</i>	Radius for the rounded corners
in	<i>nCol</i>	Color RGB value to fill

**Returns**

none

### 7.3.2.6 gslc\_DrawFillSector()

```
void gslc_DrawFillSector (
    gslc_tsGui * pGui,
    int16_t nQuality,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nRad1,
    int16_t nRad2,
    gslc_tsColor cArc,
    int16_t nAngSecStart,
    int16_t nAngSecEnd )
```

Draw a flat filled sector of a circle with support for inner and outer radius.

- Can be used to create a ring or pie chart

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nQuality</i>	Number of segments used to depict a full circle. The higher the value, the smoother the resulting arcs. A value of 72 provides 360/72=5 degrees per segment which is a reasonable compromise between smoothness and performance.
in	<i>nMidX</i>	Midpoint X coordinate of circle
in	<i>nMidY</i>	Midpoint Y coordinate of circle
in	<i>nRad1</i>	Inner sector radius (0 for sector / pie, non-zero for ring)
in	<i>nRad2</i>	Outer sector radius. Delta from nRad1 defines ring thickness.
in	<i>cArc</i>	Color for flat fill
in	<i>nAngSecStart</i>	Angle of start of sector drawing (0 at top), measured in degrees.
in	<i>nAngSecEnd</i>	Angle of end of sector drawing (0 at top), measured in degrees.

#### Returns

none

### 7.3.2.7 gslc\_DrawFillTriangle()

```
void gslc_DrawFillTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a filled triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value for the fill

## Returns

true if success, false if error

7.3.2.8 `gslc_DrawFrameCircle()`

```
void gslc_DrawFrameCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed circle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center X coordinate
in	<i>nMidY</i>	Center Y coordinate
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value for the frame

## Returns

none

7.3.2.9 `gslc_DrawFrameQuad()`

```
void gslc_DrawFrameQuad (
    gslc_tsGui * pGui,
    gslc_tsPt * psPt,
    gslc_tsColor nCol )
```

Draw a framed quadrilateral.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>psPt</i>	Pointer to array of 4 points
in	<i>nCol</i>	Color RGB value for the frame

**Returns**

true if success, false if error

**7.3.2.10 gslc\_DrawFrameRect()**

```
void gslc_DrawFrameRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a framed rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value for the frame

**Returns**

none

**7.3.2.11 gslc\_DrawFrameRoundRect()**

```
void gslc_DrawFrameRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nRadius</i>	Radius for the rounded corners
in	<i>nCol</i>	Color RGB value for the frame



**Returns**

none

**7.3.2.12 gslc\_DrawFrameTriangle()**

```
void gslc_DrawFrameTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a framed triangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value for the frame

**Returns**

true if success, false if error

**7.3.2.13 gslc\_DrawLine()**

```
void gslc_DrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw an arbitrary line using Bresenham's algorithm.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Parameters**

in	<i>nX0</i>	X coordinate of line startpoint
in	<i>nY0</i>	Y coordinate of line startpoint
in	<i>nX1</i>	X coordinate of line endpoint
in	<i>nY1</i>	Y coordinate of line endpoint
in	<i>nCol</i>	Color RGB value for the line

**Returns**

none

**7.3.2.14 gslc\_DrawLineH()**

```
void gslc_DrawLineH (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    uint16_t nW,
    gslc_tsColor nCol )
```

Draw a horizontal line.

- Note that direction of line is in +ve X axis

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of line startpoint
in	<i>nY</i>	Y coordinate of line startpoint
in	<i>nW</i>	Width of line (in +X direction)
in	<i>nCol</i>	Color RGB value for the line

**Returns**

none

**7.3.2.15 gslc\_DrawLinePolar()**

```
void gslc_DrawLinePolar (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
```

```

uint16_t nRadStart,
uint16_t nRadEnd,
int16_t n64Ang,
gslc_tsColor nCol )

```

Draw a polar ray segment.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of line startpoint
in	<i>nY</i>	Y coordinate of line startpoint
in	<i>nRadStart</i>	Starting radius of line
in	<i>nRadEnd</i>	Ending radius of line
in	<i>n64Ang</i>	Angle of ray (degrees * 64). 0 is up, +90*64 is to right From -180*64 to +180*64
in	<i>nCol</i>	Color RGB value for the line

#### Returns

none

#### 7.3.2.16 gslc\_DrawLineV()

```

void gslc_DrawLineV (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    uint16_t nH,
    gslc_tsColor nCol )

```

Draw a vertical line.

- Note that direction of line is in +ve Y axis

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of line startpoint
in	<i>nY</i>	Y coordinate of line startpoint
in	<i>nH</i>	Height of line (in +Y direction)
in	<i>nCol</i>	Color RGB value for the line

#### Returns

none

### 7.3.2.17 gslc\_DrawSetPixel()

```
void gslc_DrawSetPixel (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Set a pixel on the active screen to the given color with lock.

- Calls upon `gslc_DrvDrawSetPixelRaw()` but wraps with a surface lock lock
- If repeated access is needed, use `gslc_DrvDrawSetPixelRaw()` instead

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	Pixel X coordinate to set
in	<i>nY</i>	Pixel Y coordinate to set
in	<i>nCol</i>	Color pixel value to assign

#### Returns

none

## 7.4 Font Functions

Functions that load fonts.

### Functions

- `bool gslc_FontAdd (gslc_tsGui *pGui, int16_t nFontId, gslc_tFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)`  
Load a font into the local font cache and assign font ID (nFontId).
- `bool gslc_FontSet (gslc_tsGui *pGui, int16_t nFontId, gslc_tFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)`  
Load a font into the local font cache and store as font ID (nFontId)
- `gslc_tsFont * gslc_FontGet (gslc_tsGui *pGui, int16_t nFontId)`  
Fetch a font from its ID value.
- `bool gslc_FontSetMode (gslc_tsGui *pGui, int16_t nFontId, gslc_tFontRefMode eFontMode)`  
Set the font operating mode.

### 7.4.1 Detailed Description

Functions that load fonts.

### 7.4.2 Function Documentation

#### 7.4.2.1 gslc\_FontAdd()

```
bool gslc_FontAdd (
    gslc_tsGui * pGui,
    int16_t nFontId,
    gslc_tFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font into the local font cache and assign font ID (nFontId).

- Font is stored into next available internal array element
- NOTE: Use FontSet() instead

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nFontId</i>	ID to use when referencing this font
in	<i>eFontRefType</i>	Font reference type (eg. filename or pointer)
in	<i>pvFontRef</i>	Reference pointer to identify the font. In the case of SDL mode, it is a filepath to the font file. In the case of Arduino it is a pointer value to the font bitmap array (GFXFont)
in	<i>nFontSz</i>	Typeface size to use (only used in SDL mode)

**Returns**

true if load was successful, false otherwise

**7.4.2.2 gslc\_FontGet()**

```
gslc_tsFont* gslc_FontGet (
    gslc_tsGui * pGui,
    int16_t nFontId )
```

Fetch a font from its ID value.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nFontId</i>	ID value used to reference the font (supplied originally to <a href="#">gslc_FontAdd()</a> )

**Returns**

A pointer to the font structure or NULL if error

**7.4.2.3 gslc\_FontSet()**

```
bool gslc_FontSet (
    gslc_tsGui * pGui,
    int16_t nFontId,
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font into the local font cache and store as font ID (nFontId)

- Font is stored into index nFontId, so nFontId must be from separate font enum (0-based).
- Example: enum { E\_FONT\_BTN, E\_FONT\_TXT, MAX\_FONT };

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nFontId</i>	ID to use when referencing this font
in	<i>eFontRefType</i>	Font reference type (eg. filename or pointer)
in	<i>pvFontRef</i>	Reference pointer to identify the font. In the case of SDL mode, it is a filepath to the font file. In the case of Arduino it is a pointer value to the font bitmap array (GFXFont)
in	<i>nFontSz</i>	Typeface size to use (only used in SDL mode)

**Returns**

true if load was successful, false otherwise

**7.4.2.4 gslc\_FontSetMode()**

```
bool gslc_FontSetMode (
    gslc_tsGui * pGui,
    int16_t nFontId,
    gslc_teFontRefMode eFontMode )
```

Set the font operating mode.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nFontId</i>	ID value used to reference the font (supplied originally to <a href="#">gslc_FontAdd()</a> )
in, out	<i>eFontMode</i>	Font mode to assign to this font

**Returns**

true if success

## 7.5 Page Functions

Functions that operate at the page level.

### Functions

- `int gslc_GetPageCur (gslc_tsGui *pGui)`  
*Fetch the current page ID.*
- `void gslc_SetStackPage (gslc_tsGui *pGui, uint8_t nStackPos, int16_t nPageId)`  
*Assign a page to the page stack.*
- `void gslc_SetStackState (gslc_tsGui *pGui, uint8_t nStackPos, bool bActive, bool bDoDraw)`  
*Change the status of a page in a page stack.*
- `void gslc_SetPageBase (gslc_tsGui *pGui, int16_t nPageId)`  
*Assigns a page for the base layer in the page stack.*
- `void gslc_SetPageCur (gslc_tsGui *pGui, int16_t nPageId)`  
*Select a page for the current layer in the page stack.*
- `void gslc_SetPageOverlay (gslc_tsGui *pGui, int16_t nPageId)`  
*Select a page for the overlay layer in the page stack.*
- `void gslc_PopupShow (gslc_tsGui *pGui, int16_t nPageId, bool bModal)`  
*Show a popup dialog.*
- `void gslc_PopupHide (gslc_tsGui *pGui)`  
*Hides the currently active popup dialog.*
- `void gslc_PageRedrawSet (gslc_tsGui *pGui, bool bRedraw)`  
*Update the need-redraw status for the current page.*
- `bool gslc_PageRedrawGet (gslc_tsGui *pGui)`  
*Get the need-redraw status for the current page.*
- `void gslc_PageAdd (gslc_tsGui *pGui, int16_t nPageId, gslc_tsElem *psElem, uint16_t nMaxElem, gslc_tsElemRef *psElemRef, uint16_t nMaxElemRef)`  
*Add a page to the GUI.*
- `gslc_tsElemRef * gslc_PageFindElemById (gslc_tsGui *pGui, int16_t nPageId, int16_t nElemId)`  
*Find an element in the GUI by its Page ID and Element ID.*

### 7.5.1 Detailed Description

Functions that operate at the page level.

### 7.5.2 Function Documentation

#### 7.5.2.1 `gslc_GetPageCur()`

```
int gslc_GetPageCur (
    gslc_tsGui * pGui )
```

Fetch the current page ID.



## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

Page ID

## 7.5.2.2 gslc\_PageAdd()

```
void gslc_PageAdd (
    gslc_tsGui * pGui,
    int16_t nPageId,
    gslc_tsElem * psElem,
    uint16_t nMaxElem,
    gslc_tsElemRef * psElemRef,
    uint16_t nMaxElemRef )
```

Add a page to the GUI.

- This call associates an element array with the collection within the page
- Once a page has been added to the GUI, elements can be added to the page by specifying the same page ID

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageId</i>	Page ID to assign
in	<i>psElem</i>	Internal element array storage to associate with the page
in	<i>nMaxElem</i>	Maximum number of elements that can be added to the internal element array (ie. RAM))
in	<i>psElemRef</i>	Internal element reference array storage to associate with the page. All elements, whether they are located in the internal element array or in external Flash (PROGMEM) storage, require an entry in the element reference array.
in	<i>nMaxElemRef</i>	Maximum number of elements in the reference array. This is effectively the maximum number of elements that can appear on a page, irrespective of whether it is stored in RAM or Flash (PROGMEM).

## Returns

none

## 7.5.2.3 gslc\_PageFindElemById()

```
gslc_tsElemRef* gslc_PageFindElemById (
    gslc_tsGui * pGui,
```

```

    int16_t nPageId,
    int16_t nElemId )

```

Find an element in the GUI by its Page ID and Element ID.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>n</i> ↔ <i>PageId</i>	Page ID to search
in	<i>n</i> ↔ <i>ElemId</i>	Element ID to search

#### Returns

Ptr to an element or NULL if none found

#### 7.5.2.4 gslc\_PageRedrawGet()

```

bool gslc_PageRedrawGet (
    gslc_tsGui * pGui )

```

Get the need-redraw status for the current page.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

True if redraw required, false otherwise

#### 7.5.2.5 gslc\_PageRedrawSet()

```

void gslc_PageRedrawSet (
    gslc_tsGui * pGui,
    bool bRedraw )

```

Update the need-redraw status for the current page.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>bRedraw</i>	True if redraw required, false otherwise

**Returns**

none

**7.5.2.6 gslc\_PopupHide()**

```
void gslc_PopupHide (
    gslc_tsGui * pGui )
```

Hides the currently active popup dialog.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**7.5.2.7 gslc\_PopupShow()**

```
void gslc_PopupShow (
    gslc_tsGui * pGui,
    int16_t nPageId,
    bool bModal )
```

Show a popup dialog.

- Popup dialogs use the overlay layer in the page stack

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔PageId</i>	Page ID to use as the popup dialog
in	<i>bModal</i>	If true, popup is modal (other layers won't accept touch). If false, popup is modeless (other layers still accept touch)

**Returns**

none

### 7.5.2.8 gslc\_SetPageBase()

```
void gslc_SetPageBase (
    gslc_tsGui * pGui,
    int16_t nPageId )
```

Assigns a page for the base layer in the page stack.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageId</i>	Page ID to select (or GSLC_PAGE_NONE to disable)

#### Returns

none

### 7.5.2.9 gslc\_SetPageCur()

```
void gslc_SetPageCur (
    gslc_tsGui * pGui,
    int16_t nPageId )
```

Select a page for the current layer in the page stack.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageId</i>	Page ID to select

#### Returns

none

### 7.5.2.10 gslc\_SetPageOverlay()

```
void gslc_SetPageOverlay (
    gslc_tsGui * pGui,
    int16_t nPageId )
```

Select a page for the overlay layer in the page stack.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageId</i>	Page ID to select (or GSLC_PAGE_NONE to disable)

## Returns

none

## 7.5.2.11 gslc\_SetStackPage()

```
void gslc_SetStackPage (
    gslc_tsGui * pGui,
    uint8_t nStackPos,
    int16_t nPageId )
```

Assign a page to the page stack.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nStackPos</i>	Position to update in the page stack (0..GSLC_STACK__MAX-1)
in	<i>nPageId</i>	Page ID to select as current

## Returns

none

## 7.5.2.12 gslc\_SetStackState()

```
void gslc_SetStackState (
    gslc_tsGui * pGui,
    uint8_t nStackPos,
    bool bActive,
    bool bDoDraw )
```

Change the status of a page in a page stack.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nStackPos</i>	Position to update in the page stack (0..GSLC_STACK__MAX-1)
in	<i>bActive</i>	Indicate if page should receive touch events
in	<i>bDoDraw</i>	Indicate if page should continue to be redrawn. If pages in the stack are overlapping and an element in a lower layer continues to receive updates, then the element may "show through" the layers above it. In such cases where pages in the stack are overlapping and lower pages contain dynamically updating elements, it may be best to disable redraw while the overlapping page is visible (by setting bDoDraw to false).
Generated by Doxygen		

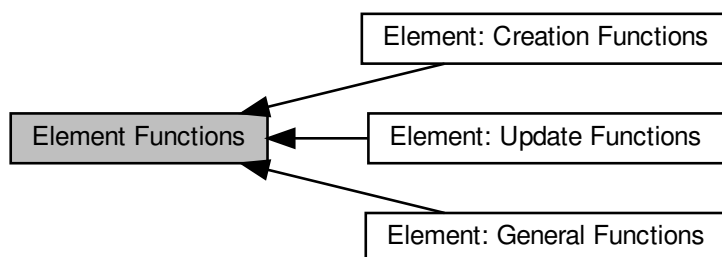
**Returns**

none

## 7.6 Element Functions

Functions that are used to create and manipulate elements.

Collaboration diagram for Element Functions:



### Modules

- [Element: Creation Functions](#)  
*Functions that create GUI elements.*
- [Element: General Functions](#)  
*General-purpose functions that operate on Elements.*
- [Element: Update Functions](#)  
*Functions that configure or modify an existing element.*

### 7.6.1 Detailed Description

Functions that are used to create and manipulate elements.

## 7.7 Element: Creation Functions

Functions that create GUI elements.

Collaboration diagram for Element: Creation Functions:



### Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create a Text Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId, [GSLC\\_CB\\_TOUCH](#) cbTouch)  
*Create a textual Button Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, [gslc\\_tsImgRef](#) sImgRef, [gslc\\_tsImgRef](#) sImgRefSel, [GSLC\\_CB\\_TOUCH](#) cbTouch)  
*Create a graphical Button Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBox](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) r↔Elem)  
*Create a Box Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1)  
*Create a Line Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) r↔Elem, [gslc\\_tsImgRef](#) sImgRef)  
*Create an image Element.*

### 7.7.1 Detailed Description

Functions that create GUI elements.

### 7.7.2 Function Documentation



7.7.2.1 `gslc_ElemCreateBox()`

```
gslc_tsElemRef* gslc_ElemCreateBox (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsRect rElem )
```

Create a Box Element.

- Draws a box with frame and fill

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>rElem</i>	Rectangle coordinates defining box size

**Returns**

Pointer to the Element reference or NULL if failure

**7.7.2.2 gslc\_ElemCreateBtnImg()**

```
gslc_tsElemRef* gslc_ElemCreateBtnImg (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsRect rElem,
    gslc_tsImgRef sImgRef,
    gslc_tsImgRef sImgRefSel,
    GSLC_CB_TOUCH cbTouch )
```

Create a graphical Button Element.

- Creates a clickable element that uses a BMP image with no frame or fill
- Transparency is supported by bitmap color (0xFF00FF)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>rElem</i>	Rectangle coordinates defining image size
in	<i>sImgRef</i>	Image reference to load (unselected state)
in	<i>sImgRefSel</i>	Image reference to load (selected state)
in	<i>cbTouch</i>	Callback for touch events

**Returns**

Pointer to the Element reference or NULL if failure

7.7.2.3 `gslc_ElemCreateBtnTxt()`

```
gslc_tsElemRef* gslc_ElemCreateBtnTxt (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId,
    GSLC_CB_TOUCH cbTouch )
```

Create a textual Button Element.

- Creates a clickable element that has a textual label with frame and fill

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>rElem</i>	Rectangle coordinates defining text background size
in	<i>pStrBuf</i>	String to copy into element
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf). Only applicable if GSLC_LOCAL_STR=0. Ignored if GSLC_LOCAL_STR=1.)
in	<i>nFontId</i>	Font ID to use for text display
in	<i>cbTouch</i>	Callback for touch events

## Returns

Pointer to the Element reference or NULL if failure

7.7.2.4 `gslc_ElemCreateImg()`

```
gslc_tsElemRef* gslc_ElemCreateImg (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsRect rElem,
    gslc_tsImgRef sImgRef )
```

Create an image Element.

- Draws an image

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>rElem</i>	Rectangle coordinates defining box size
in	<i>sImgRef</i>	Image reference to load

**Returns**

Pointer to the Element reference or NULL if failure

**7.7.2.5 gslc\_ElemCreateLine()**

```
gslc_tsElemRef* gslc_ElemCreateLine (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1 )
```

Create a Line Element.

- Draws a line with fill color

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX0</i>	X coordinate of line startpoint
in	<i>nY0</i>	Y coordinate of line startpoint
in	<i>nX1</i>	X coordinate of line endpoint
in	<i>nY1</i>	Y coordinate of line endpoint

**Returns**

Pointer to the Element reference or NULL if failure

## 7.7.2.6 gslc\_ElemCreateTxt()

```

gslc_tsElemRef* gslc_ElemCreateTxt (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )

```

Create a Text Element.

- Draws a text string with filled background

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>rElem</i>	Rectangle coordinates defining text background size
in	<i>pStrBuf</i>	String to copy into element
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf). Only applicable if GSLC_LOCAL_STR=0. Ignored if GSLC_LOCAL_STR=1.)
in	<i>nFontId</i>	Font ID to use for text display

## Returns

Pointer to the Element reference or NULL if failure

## 7.8 Element: General Functions

General-purpose functions that operate on Elements.

Collaboration diagram for Element: General Functions:



### Functions

- int [gslc\\_ElemGetId](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get an Element ID from an element structure.*

#### 7.8.1 Detailed Description

General-purpose functions that operate on Elements.

#### 7.8.2 Function Documentation

##### 7.8.2.1 [gslc\\_ElemGetId](#)()

```
int gslc_ElemGetId (
    gslc\_tsGui * pGui,
    gslc\_tsElemRef * pElemRef )
```

Get an Element ID from an element structure.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference structure

##### Returns

ID of element or GSLC\_ID\_NONE if not found

## 7.9 Element: Update Functions

Functions that configure or modify an existing element.

Collaboration diagram for Element: Update Functions:



### Functions

- void `gslc_ElemSetFillEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, bool bFillEn)  
*Set the fill state for an Element.*
- void `gslc_ElemSetFrameEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, bool bFrameEn)  
*Set the frame state for an Element.*
- void `gslc_ElemSetRoundEn` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, bool bRoundEn)  
*Set the rounded frame/fill state for an Element.*
- void `gslc_ElemSetCol` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `gslc_tsColor` colFrame, `gslc_tsColor` colFill, `gslc_tsColor` colFillGlow)  
*Update the common color selection for an Element.*
- void `gslc_ElemSetGlowCol` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `gslc_tsColor` colFrameGlow, `gslc_tsColor` colFillGlow, `gslc_tsColor` colTxtGlow)  
*Update the common color selection for glowing state of an Element.*
- void `gslc_ElemSetGroup` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, int nGroupId)  
*Set the group ID for an element.*
- int `gslc_ElemGetGroup` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)  
*Get the group ID for an element.*
- void `gslc_ElemSetRect` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `gslc_tsRect` rElem)  
*Set the position and size for an element.*
- `gslc_tsRect` `gslc_ElemGetRect` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)  
*Get the rectangular region for an element.*
- void `gslc_ElemSetTxtAlign` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, unsigned nAlign)  
*Set the alignment of a textual element (horizontal and vertical)*
- void `gslc_ElemSetTxtMargin` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, unsigned nMargin)  
*Set the margin around of a textual element.*
- void `gslc_ElemSetTxtMarginXY` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, int8\_t nMarginX, int8\_t nMarginY)  
*Set the margin around of a textual element (X & Y offsets can be different)*
- void `gslc_StrCopy` (char \*pDstStr, const char \*pSrcStr, uint16\_t nDstLen)  
*Helper routine to perform string deep copy.*
- void `gslc_ElemSetTxtStr` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, const char \*pStr)  
*Update the text string associated with an Element.*
- char \* `gslc_ElemGetTxtStr` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)  
*Fetch the current text string associated with an Element.*

- void [gslc\\_ElemSetTxtCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colVal)  
*Update the text string color associated with an Element ID.*
- void [gslc\\_ElemSetTxtMem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)  
*Update the text string location in memory.*
- void [gslc\\_ElemSetTxtEnc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)  
*Update the text string encoding mode.*
- void [gslc\\_ElemUpdateFont](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int nFontId)  
*Update the Font selected for an Element's text.*
- void [gslc\\_ElemSetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Update the need-redraw status for an element.*
- [gslc\\_teRedrawType](#) [gslc\\_ElemGetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the need-redraw status for an element.*
- void [gslc\\_ElemSetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowEn)  
*Update the glowing enable for an element.*
- void [gslc\\_ElemSetClickEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bClickEn)  
*Update the click enable for an element.*
- void [gslc\\_ElemSetTouchFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_TOUCH](#) funcCb)  
*Update the touch function callback for an element.*
- void [gslc\\_ElemSetStyleFrom](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRefSrc, [gslc\\_tsElemRef](#) \*pElemRefDest)  
*Copy style settings from one element to another.*
- void [gslc\\_ResetRectState](#) ([gslc\\_tsRectState](#) \*pState)  
*Reset the element region state struct.*
- void [gslc\\_ElemCalcRectState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRectState](#) \*pState)  
*Calculate the element region state struct.*
- int8\_t [gslc\\_ElemCalcResizeForFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Calculate the change in dimensions of an element to account for any change in focus and/or frame attributes.*
- void [gslc\\_ElemGrowRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nDelta)  
*Increase or decrease the size of an element's region.*
- bool [gslc\\_ElemGetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the glowing enable for an element.*
- void [gslc\\_ElemSetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowing)  
*Update the glowing indicator for an element.*
- bool [gslc\\_ElemGetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the glowing indicator for an element.*
- bool [gslc\\_ElemGetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the focus enable for an element.*
- void [gslc\\_ElemSetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocusEn)  
*Set the focus enable for an element.*
- void [gslc\\_ElemSetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocused)  
*Update the focused indicator for an element.*
- bool [gslc\\_ElemGetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the focused indicator for an element.*
- void [gslc\\_ElemSetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bEditing)  
*Update the editing indicator for an element.*
- bool [gslc\\_ElemGetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the editing indicator for an element.*
- void [gslc\\_ElemSetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bVisible)  
*Update the visibility status for an element.*
- bool [gslc\\_ElemGetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the visibility status for an element.*



- bool `gslc_ElemGetOnScreen` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef)  
*Determine whether an element is visible on the screen.*
- void `gslc_ElemSetDrawFunc` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `GSLC_CB_DRAW` funcCb)  
*Assign the drawing callback function for an element.*
- void `gslc_ElemSetTickFunc` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `GSLC_CB_TICK` funcCb)  
*Assign the tick callback function for an element.*
- bool `gslc_ElemOwnsCoord` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, int16\_t nX, int16\_t nY, bool b↵ OnlyClickEn)  
*Determine if a coordinate is inside of an element.*

### 7.9.1 Detailed Description

Functions that configure or modify an existing element.

### 7.9.2 Function Documentation

#### 7.9.2.1 `gslc_ElemCalcRectState()`

```
void gslc_ElemCalcRectState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsRectState * pState )
```

Calculate the element region state struct.

- Establishes the size of the frame and inner regions
- Determines the color of various parts of the element (focus rect, frame, text, fill, etc.)
- The region state is calculated based upon the element's attributes, including the frame enable, ability to support focus, whether the contents can be shrunk to accommodate a frame, and the current state of focus/edit/etc.

#### Parameters

in	<code>pGui</code>	Pointer to GUI
in	<code>pElemRef</code>	Pointer to Element reference
out	<code>pState</code>	Pointer to element region state

#### Returns

none

### 7.9.2.2 gslc\_ElemCalcResizeForFocus()

```
int8_t gslc_ElemCalcResizeForFocus (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Calculate the change in dimensions of an element to account for any change in focus and/or frame attributes.

It also takes into account the "NoShrink" attribute which indicates that an element's contents can't be reduced in size (eg. because they contain a fixed-sized image).

- This routine is usually called whenever an element is created and also whenever size-impacting features are adjusted (eg. ElemSetFrameEn).

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

The increase (or decrease) in size of the element

### 7.9.2.3 gslc\_ElemGetEdit()

```
bool gslc_ElemGetEdit (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the editing indicator for an element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

True if element is being edited

### 7.9.2.4 gslc\_ElemGetFocus()

```
bool gslc_ElemGetFocus (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the focused indicator for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

True if element is focused

7.9.2.5 `gslc_ElemGetFocusEn()`

```
bool gslc_ElemGetFocusEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the focus enable for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

True if element supports focus

**Todo** Doc

7.9.2.6 `gslc_ElemGetGlow()`

```
bool gslc_ElemGetGlow (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the glowing indicator for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

True if element is glowing

**7.9.2.7 gslc\_ElemGetGlowEn()**

```
bool gslc_ElemGetGlowEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the glowing enable for an element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

True if element supports glowing

**7.9.2.8 gslc\_ElemGetGroup()**

```
int gslc_ElemGetGroup (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the group ID for an element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Group ID or GSLC\_GROUP\_ID\_NONE if unassigned

**7.9.2.9 gslc\_ElemGetOnScreen()**

```
bool gslc_ElemGetOnScreen (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Determine whether an element is visible on the screen.

- This function takes into account both the element's "Visible" state as well as whether the element's associated page is active in the page stack.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

True if element appears on the screen, false otherwise

**7.9.2.10 gslc\_ElemGetRect()**

```
gslc_tsRect gslc_ElemGetRect (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the rectangular region for an element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Rect region of an element

**7.9.2.11 gslc\_ElemGetRedraw()**

```
gslc_teRedrawType gslc_ElemGetRedraw (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the need-redraw status for an element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Redraw status

**7.9.2.12 gslc\_ElemGetTxtStr()**

```
char* gslc_ElemGetTxtStr (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Fetch the current text string associated with an Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Pointer to character array string

**7.9.2.13 gslc\_ElemGetVisible()**

```
bool gslc_ElemGetVisible (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the visibility status for an element.

- Note that the visibility state is independent of whether or not the page associated with the element is actively displayed.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

True if element is marked as visible, false if hidden

## 7.9.2.14 gslc\_ElemGrowRect()

```
void gslc_ElemGrowRect (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nDelta )
```

Increase or decrease the size of an element's region.

- Update the redraw status as needed

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nDelta</i>	The increase of element size (negative for decrease)

## Returns

none

## 7.9.2.15 gslc\_ElemOwnsCoord()

```
bool gslc_ElemOwnsCoord (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nX,
    int16_t nY,
    bool bOnlyClickEn )
```

Determine if a coordinate is inside of an element.

- This routine is useful in determining if a touch coordinate is inside of a button.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Element reference used for boundary test
in	<i>nX</i>	X coordinate to test
in	<i>nY</i>	Y coordinate to test
in	<i>bOnlyClickEn</i>	Only output true if element was also marked as "clickable" (eg. bClickEn=true)

## Returns

true if inside element, false otherwise

### 7.9.2.16 gslc\_ElemSetClickEn()

```
void gslc_ElemSetClickEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bClickEn )
```

Update the click enable for an element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bClickEn</i>	True if element should support click events

#### Returns

none

### 7.9.2.17 gslc\_ElemSetCol()

```
void gslc_ElemSetCol (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colFrame,
    gslc_tsColor colFill,
    gslc_tsColor colFillGlow )
```

Update the common color selection for an Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>colFillGlow</i>	Color for the fill when glowing

#### Returns

none

### 7.9.2.18 gslc\_ElemSetDrawFunc()

```
void gslc_ElemSetDrawFunc (
    gslc_tsGui * pGui,
```



```
gslc_tsElemRef * pElemRef,
GSLC_CB_DRAW funcCb )
```

Assign the drawing callback function for an element.

- This allows the user to override the default rendering for an element, enabling the creation of a custom element

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to drawing routine (or NULL for default))

#### Returns

none

#### 7.9.2.19 gslc\_ElemSetEdit()

```
void gslc_ElemSetEdit (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bEditing )
```

Update the editing indicator for an element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bEditing</i>	True if element is being edited

#### Returns

none

#### 7.9.2.20 gslc\_ElemSetFillEn()

```
void gslc_ElemSetFillEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFillEn )
```

Set the fill state for an Element.

- If not filled, the element can support transparency against an arbitrary background, but this can require full screen redraws if the element is updated.
- If filled, the background fill color can be changed by [gslc\\_ElemSetCol\(\)](#)

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFillEn</i>	True if filled, false otherwise

#### Returns

none

#### 7.9.2.21 gslc\_ElemSetFocus()

```
void gslc_ElemSetFocus (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFocused )
```

Update the focused indicator for an element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFocused</i>	True if element is focused

#### Returns

none

#### 7.9.2.22 gslc\_ElemSetFocusEn()

```
void gslc_ElemSetFocusEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFocusEn )
```

Set the focus enable for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFocusEn</i>	Enable focus if 1, 0 if not supported

## Returns

none

**Todo** Doc

## 7.9.2.23 gslc\_ElemSetFrameEn()

```
void gslc_ElemSetFrameEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFrameEn )
```

Set the frame state for an Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFrameEn</i>	True if framed, false otherwise

## Returns

none

## 7.9.2.24 gslc\_ElemSetGlow()

```
void gslc_ElemSetGlow (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bGlowing )
```

Update the glowing indicator for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bGlowing</i>	True if element is glowing

**Returns**

none

**7.9.2.25 gslc\_ElemSetGlowCol()**

```
void gslc_ElemSetGlowCol (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colFrameGlow,
    gslc_tsColor colFillGlow,
    gslc_tsColor colTxtGlow )
```

Update the common color selection for glowing state of an Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colFrameGlow</i>	Color for the frame when glowing
in	<i>colFillGlow</i>	Color for the fill when glowing
in	<i>colTxtGlow</i>	Color for the text when glowing

**Returns**

none

**7.9.2.26 gslc\_ElemSetGlowEn()**

```
void gslc_ElemSetGlowEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bGlowEn )
```

Update the glowing enable for an element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bGlowEn</i>	True if element should support glowing

**Returns**

none

### 7.9.2.27 gslc\_ElemSetGroup()

```
void gslc_ElemSetGroup (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int nGroupId )
```

Set the group ID for an element.

- Typically used to associate radio button elements together

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nGroupId</i>	Group ID to assign

#### Returns

none

### 7.9.2.28 gslc\_ElemSetRect()

```
void gslc_ElemSetRect (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsRect rElem )
```

Set the position and size for an element.

- This updates the element's rectangular region, which can be used to relocate or resize an element at runtime

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>rElem</i>	Rect region (top-left coord, width, height)

#### Returns

none

### 7.9.2.29 gslc\_ElemSetRedraw()

```
void gslc_ElemSetRedraw (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Update the need-redraw status for an element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>eRedraw</i>	Redraw state to set

#### Returns

none

### 7.9.2.30 gslc\_ElemSetRoundEn()

```
void gslc_ElemSetRoundEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bRoundEn )
```

Set the rounded frame/fill state for an Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bRoundEn</i>	True if rounded, false otherwise

#### Returns

none

### 7.9.2.31 gslc\_ElemSetStyleFrom()

```
void gslc_ElemSetStyleFrom (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRefSrc,
    gslc_tsElemRef * pElemRefDest )
```

Copy style settings from one element to another.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRefSrc</i>	Pointer to source Element reference
in	<i>pElemRefDest</i>	Pointer to destination Element reference

## Returns

none

7.9.2.32 `gslc_ElemSetTickFunc()`

```
void gslc_ElemSetTickFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_TICK funcCb )
```

Assign the tick callback function for an element.

- This allows the user to provide background updates to an element triggered by the main loop call to `gslc_↵ Update()`

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to tick routine (or NULL for none)

## Returns

none

7.9.2.33 `gslc_ElemSetTouchFunc()`

```
void gslc_ElemSetTouchFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_TOUCH funcCb )
```

Update the touch function callback for an element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Pointer to the touch callback function

**Returns**

none

**7.9.2.34 gslc\_ElemSetTxtAlign()**

```
void gslc_ElemSetTxtAlign (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    unsigned nAlign )
```

Set the alignment of a textual element (horizontal and vertical)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nAlign</i>	Alignment to specify: <ul style="list-style-type: none"> <li>• GSLC_ALIGN_TOP_LEFT</li> <li>• GSLC_ALIGN_TOP_MID</li> <li>• GSLC_ALIGN_TOP_RIGHT</li> <li>• GSLC_ALIGN_MID_LEFT</li> <li>• GSLC_ALIGN_MID_MID</li> <li>• GSLC_ALIGN_MID_RIGHT</li> <li>• GSLC_ALIGN_BOT_LEFT</li> <li>• GSLC_ALIGN_BOT_MID</li> <li>• GSLC_ALIGN_BOT_RIGHT</li> </ul>

**Returns**

none

**7.9.2.35 gslc\_ElemSetTxtCol()**

```
void gslc_ElemSetTxtCol (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colVal )
```

Update the text string color associated with an Element ID.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colVal</i>	RGB color to change to

## Returns

none

## 7.9.2.36 gslc\_ElemSetTxtEnc()

```
void gslc_ElemSetTxtEnc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teTxtFlags eFlags )
```

Update the text string encoding mode.

- This function can be used to indicate that the element's text string is encoded in UTF-8, which supports extended / foreign character maps

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>eFlags</i>	Flags associated with text encoding (GSLC_TXT_ENC_*)

## Returns

none

## 7.9.2.37 gslc\_ElemSetTxtMargin()

```
void gslc_ElemSetTxtMargin (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    unsigned nMargin )
```

Set the margin around of a textual element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nMargin</i>	Number of pixels gap to leave surrounding text

**Returns**

none

**7.9.2.38 gslc\_ElemSetTxtMarginXY()**

```
void gslc_ElemSetTxtMarginXY (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nMarginX,
    int8_t nMarginY )
```

Set the margin around of a textual element (X & Y offsets can be different)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nMarginX</i>	Number of pixels gap to offset text horizontally
in	<i>nMarginY</i>	Number of pixels gap to offset text vertically

**Returns**

none

**7.9.2.39 gslc\_ElemSetTxtMem()**

```
void gslc_ElemSetTxtMem (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teTxtFlags eFlags )
```

Update the text string location in memory.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>eFlags</i>	Flags associated with text memory location (GSLC_TXT_MEM_*)

**Returns**

none

#### 7.9.2.40 gslc\_ElemSetTxtStr()

```
void gslc_ElemSetTxtStr (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    const char * pStr )
```

Update the text string associated with an Element.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>pStr</i>	String to copy into element

##### Returns

none

#### 7.9.2.41 gslc\_ElemSetVisible()

```
void gslc_ElemSetVisible (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bVisible )
```

Update the visibility status for an element.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bVisible</i>	True if element is shown, false if hidden

##### Returns

none

#### 7.9.2.42 gslc\_ElemUpdateFont()

```
void gslc_ElemUpdateFont (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int nFontId )
```

Update the Font selected for an Element's text.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nFontId</i>	Font ID to select

**Returns**

none

**7.9.2.43 gslc\_ResetRectState()**

```
void gslc_ResetRectState (
    gslc_tsRectState * pState )
```

Reset the element region state struct.

**Parameters**

out	<i>pState</i>	Pointer to the element region structure
-----	---------------	---

**Returns**

none

**7.9.2.44 gslc\_StrCopy()**

```
void gslc_StrCopy (
    char * pDstStr,
    const char * pSrcStr,
    uint16_t nDstLen )
```

Helper routine to perform string deep copy.

- Includes termination
- Similar to strncpy() except:
  - nDstLen is the total buffer size (including terminator)
  - A terminator is added at the end of the buffer

**Parameters**

in, out	<i>pDstStr</i>	Pointer to destination buffer
in	<i>nDstLen</i>	Size of destination buffer (includes NULL)
in	<i>pSrcStr</i>	Pointer to source buffer

**Returns**

none

## 7.10 Touchscreen Functions

Functions that configure and respond to a touch device.

### Macros

- `#define TOUCH_ROTATION_DATA`  
*Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.*
- `#define TOUCH_ROTATION_DATA`  
*Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.*
- `#define TOUCH_ROTATION_SWAPXY(rotation)`
- `#define TOUCH_ROTATION_SWAPXY(rotation)`
- `#define TOUCH_ROTATION_FLIPX(rotation)`
- `#define TOUCH_ROTATION_FLIPX(rotation)`
- `#define TOUCH_ROTATION_FLIPY(rotation)`
- `#define TOUCH_ROTATION_FLIPY(rotation)`

### Functions

- `bool gslc_InitTouch (gslc_tsGui *pGui, const char *acDev)`  
*Initialize the touchscreen device driver.*
- `bool gslc_GetTouch (gslc_tsGui *pGui, int16_t *pnX, int16_t *pnY, uint16_t *pnPress, gslc_telInputRawEvent *peInputEvent, int16_t *pnInputVal)`  
*Initialize the touchscreen device driver.*
- `void gslc_SetTouchRemapEn (gslc_tsGui *pGui, bool bEn)`  
*Configure touchscreen remapping.*
- `void gslc_SetTouchRemapCal (gslc_tsGui *pGui, uint16_t nXMin, uint16_t nXMax, uint16_t nYMin, uint16_t nYMax)`  
*Configure touchscreen calibration remapping values.*
- `void gslc_SetTouchPressCal (gslc_tsGui *pGui, uint16_t nPressMin, uint16_t nPressMax)`  
*Configure touchscreen calibration pressure values.*
- `void gslc_SetTouchRemapYX (gslc_tsGui *pGui, bool bSwap)`  
*Configure touchscreen XY swap.*
- `void gslc_SetTouchEn (gslc_tsGui *pGui, bool bEn)`  
*Make touchscreen sensitive (GUI reacts to touch events) or insensitive (GUI ignores touch events)*
- `bool gslc_GetTouchEn (gslc_tsGui *pGui)`  
*Get whether the GUI will react to touch events or not.*

#### 7.10.1 Detailed Description

Functions that configure and respond to a touch device.

#### 7.10.2 Macro Definition Documentation

**7.10.2.1 TOUCH\_ROTATION\_DATA** [1/2]

```
#define TOUCH_ROTATION_DATA
```

Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.

**7.10.2.2 TOUCH\_ROTATION\_DATA** [2/2]

```
#define TOUCH_ROTATION_DATA
```

Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.

**7.10.2.3 TOUCH\_ROTATION\_FLIPX** [1/2]

```
#define TOUCH_ROTATION_FLIPX(  
    rotation )
```

**7.10.2.4 TOUCH\_ROTATION\_FLIPX** [2/2]

```
#define TOUCH_ROTATION_FLIPX(  
    rotation )
```

**7.10.2.5 TOUCH\_ROTATION\_FLIPY** [1/2]

```
#define TOUCH_ROTATION_FLIPY(  
    rotation )
```

**7.10.2.6 TOUCH\_ROTATION\_FLIPY** [2/2]

```
#define TOUCH_ROTATION_FLIPY(  
    rotation )
```

### 7.10.2.7 TOUCH\_ROTATION\_SWAPXY [1/2]

```
#define TOUCH_ROTATION_SWAPXY(  
    rotation )
```

### 7.10.2.8 TOUCH\_ROTATION\_SWAPXY [2/2]

```
#define TOUCH_ROTATION_SWAPXY(  
    rotation )
```

## 7.10.3 Function Documentation

### 7.10.3.1 gslc\_GetTouch()

```
bool gslc_GetTouch (
    gslc_tsGui * pGui,
    int16_t * pnX,
    int16_t * pnY,
    uint16_t * pnPress,
    gslc_teInputRawEvent * peInputEvent,
    int16_t * pnInputVal )
```

Initialize the touchscreen device driver.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
out	<i>pnX</i>	Ptr to int to contain latest touch X coordinate
out	<i>pnY</i>	Ptr to int to contain latest touch Y coordinate
out	<i>pnPress</i>	Ptr to int to contain latest touch pressure value
out	<i>peInputEvent</i>	Indication of event type
out	<i>pnInputVal</i>	Additional data for event type

#### Returns

true if touch event, false otherwise

### 7.10.3.2 gslc\_GetTouchEn()

```
bool gslc_GetTouchEn (
    gslc_tsGui * pGui )
```

Get whether the GUI will react to touch events or not.



**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

whether the GUI is sensitive for touch events

**7.10.3.3 gslc\_InitTouch()**

```
bool gslc_InitTouch (
    gslc_tsGui * pGui,
    const char * acDev )
```

Initialize the touchscreen device driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen (or "" if not applicable)) eg. "/dev/input/touchscreen"

**Returns**

true if successful

**7.10.3.4 gslc\_SetTouchEn()**

```
void gslc_SetTouchEn (
    gslc_tsGui * pGui,
    bool bEn )
```

Make touchscreen sensitive (GUI reacts to touch events) or insensitive (GUI ignores touch events)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>bEn</i>	Enable GUI reaction to touch events

**Returns**

none

### 7.10.3.5 gslc\_SetTouchPressCal()

```
void gslc_SetTouchPressCal (
    gslc_tsGui * pGui,
    uint16_t nPressMin,
    uint16_t nPressMax )
```

Configure touchscreen calibration pressure values.

- Only used if calibration remapping has been enabled

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPressMin</i>	Resistive touchscreen pressure min value
in	<i>nPressMax</i>	Resistive touchscreen pressure max value

#### Returns

none

### 7.10.3.6 gslc\_SetTouchRemapCal()

```
void gslc_SetTouchRemapCal (
    gslc_tsGui * pGui,
    uint16_t nXMin,
    uint16_t nXMax,
    uint16_t nYMin,
    uint16_t nYMax )
```

Configure touchscreen calibration remapping values.

- Only used if calibration remapping has been enabled

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nXMin</i>	Resistive touchscreen X_MIN calibration value
in	<i>nXMax</i>	Resistive touchscreen X_MAX calibration value
in	<i>nYMin</i>	Resistive touchscreen Y_MIN calibration value
in	<i>nYMax</i>	Resistive touchscreen Y_MAX calibration value

#### Returns

none

### 7.10.3.7 gslc\_SetTouchRemapEn()

```
void gslc_SetTouchRemapEn (
    gslc_tsGui * pGui,
    bool bEn )
```

Configure touchscreen remapping.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>bEn</i>	Enable touchscreen remapping?

#### Returns

none

### 7.10.3.8 gslc\_SetTouchRemapYX()

```
void gslc_SetTouchRemapYX (
    gslc_tsGui * pGui,
    bool bSwap )
```

Configure touchscreen XY swap.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>bSwap</i>	Enable touchscreen XY swap

#### Returns

none

## 7.11 Input Mapping Functions

Functions that handle GPIO / pin and keyboard input.

### Functions

- void [gslc\\_SetPinPollFunc](#) ([gslc\\_tsGui](#) \*pGui, [GSLC\\_CB\\_PIN\\_POLL](#) pfunc)  
*Specify the callback function that is used to collect the state of any external inputs (eg.*
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)  
*Specify the mapping between external pin inputs (fetched by the [SetPinPollFunc\(\)](#) callback and the GUI actions.*
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_teInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) eAction, int16\_t nActionVal)  
*Add an entry into the external input mapping table.*
- [gslc\\_tsElemRef](#) \* [gslc\\_FocusElemGet](#) ([gslc\\_tsGui](#) \*pGui)  
*Find the currently focused element.*
- void [gslc\\_FocusPageStep](#) ([gslc\\_tsGui](#) \*pGui, bool bNext)  
*Advance the focus to the next page in the page stack.*
- int16\_t [gslc\\_FocusElemStep](#) ([gslc\\_tsGui](#) \*pGui, bool bNext)  
*Advance the focus to the next element in the focused page.*
- void [gslc\\_FocusElemIndSet](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageInd, int16\_t nElemInd, bool bFocus)  
*Change the focus to the indexed element on the specified page.*
- void [gslc\\_FocusSetToTrackedElem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect)  
*Change the focus to the currently-tracked element.*

### 7.11.1 Detailed Description

Functions that handle GPIO / pin and keyboard input.

### 7.11.2 Function Documentation

#### 7.11.2.1 [gslc\\_FocusElemGet\(\)](#)

```
gslc\_tsElemRef* gslc\_FocusElemGet (
    gslc\_tsGui * pGui )
```

Find the currently focused element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

Element reference of focused widget, or NULL if none

### 7.11.2.2 gslc\_FocusElemIndSet()

```
void gslc_FocusElemIndSet (
    gslc_tsGui * pGui,
    int16_t nPageInd,
    int16_t nElemInd,
    bool bFocus )
```

Change the focus to the indexed element on the specified page.

- First clear any existing focus before setting a new focus

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageInd</i>	The index of the page containing the element
in	<i>nElemInd</i>	The index of the element containing the element
in	<i>bFocus</i>	If true, enables the focus on the specified element after clearing the focus on the old element. If false, no focus is enabled after clearing any existing focus.

#### Returns

none

### 7.11.2.3 gslc\_FocusElemStep()

```
int16_t gslc_FocusElemStep (
    gslc_tsGui * pGui,
    bool bNext )
```

Advance the focus to the next element in the focused page.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>bNext</i>	Advance to next element if true, previous if false

#### Returns

none

#### 7.11.2.4 gslc\_FocusPageStep()

```
void gslc_FocusPageStep (
    gslc_tsGui * pGui,
    bool bNext )
```

Advance the focus to the next page in the page stack.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>bNext</i>	Advance to next page if true, previous if false

##### Returns

none

#### 7.11.2.5 gslc\_FocusSetToTrackedElem()

```
void gslc_FocusSetToTrackedElem (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect )
```

Change the focus to the currently-tracked element.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	The active element collection to examine

##### Returns

none

#### 7.11.2.6 gslc\_InitInputMap()

```
void gslc_InitInputMap (
    gslc_tsGui * pGui,
    gslc_tsInputMap * asInputMap,
    uint8_t nInputMapMax )
```

Specify the mapping between external pin inputs (fetched by the SetPinPollFunc() callback and the GUI actions.

- This is used to enable external controls to navigate and manipulate the GUI.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>asInputMap</i>	Pointer to the input mapping table
in	<i>nInputMapMax</i>	Total number of entries in mapping table

## Returns

none

## 7.11.2.7 gslc\_InputMapAdd()

```
void gslc_InputMapAdd (
    gslc_tsGui * pGui,
    gslc_teInputRawEvent eInputEvent,
    int16_t nInputVal,
    gslc_teAction eAction,
    int16_t nActionVal )
```

Add an entry into the external input mapping table.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>eInputEvent</i>	The event to detect
in	<i>nInputVal</i>	The value associated with the detected event
in	<i>eAction</i>	The action to take in the GUI
in	<i>nActionVal</i>	An optional parameter to associate with the GUI action

## Returns

none

## 7.11.2.8 gslc\_SetPinPollFunc()

```
void gslc_SetPinPollFunc (
    gslc_tsGui * pGui,
    GSLC_CB_PIN_POLL pfunc )
```

Specify the callback function that is used to collect the state of any external inputs (eg.

buttons, pins, encoders, etc.)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pfunc</i>	Pointer to the callback function

**Returns**

none



## 7.12 General Purpose Macros

Macros that are used throughout the GUI for debug.

### Macros

- `#define GSLC_DEBUG_PRINT(sFmt, ...)`  
*Macro to enable optional debug output.*
- `#define GSLC_DEBUG2_PRINT(sFmt, ...)`
- `#define GSLC_DEBUG_PRINT_CONST(sFmt, ...)`
- `#define GSLC_DEBUG2_PRINT_CONST(sFmt, ...)`

### 7.12.1 Detailed Description

Macros that are used throughout the GUI for debug.

### 7.12.2 Macro Definition Documentation

#### 7.12.2.1 GSLC\_DEBUG2\_PRINT

```
#define GSLC_DEBUG2_PRINT(  
    sFmt,  
    ... )
```

#### 7.12.2.2 GSLC\_DEBUG2\_PRINT\_CONST

```
#define GSLC_DEBUG2_PRINT_CONST(  
    sFmt,  
    ... )
```

#### 7.12.2.3 GSLC\_DEBUG\_PRINT

```
#define GSLC_DEBUG_PRINT(  
    sFmt,  
    ... )
```

Macro to enable optional debug output.

- Supports printf formatting via `gslc_DebugPrintf()`
- Supports storing the format string in PROGMEM
- Note that at least one variable argument must be provided to the macro after the format string. This is a limitation of the macro definition. If no parameters are needed, then simply pass 0. For example: `GSLC_DEBUG_PRINT("Loaded OK",0);`

**Parameters**

in	<i>sFmt</i>	Format string for debug message
----	-------------	---------------------------------

**7.12.2.4 GSLC\_DEBUG\_PRINT\_CONST**

```
#define GSLC_DEBUG_PRINT_CONST(  
    sFmt,  
    ... )
```

## 7.13 Flash-based Element Macros

Macros that represent element creation routines based in FLASH memory.

### Macros

- `#define gslc_ElemCreateTxt_P(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, pFont, colTxt, colFrame, colFill, nAlignTxt, bFrameEn, bFillEn)`  
*Create a read-only text element.*
- `#define gslc_ElemCreateTxt_P_R(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, strLength, pFont, colTxt, colFrame, colFill, nAlignTxt, bFrameEn, bFillEn)`  
*Create a read-write text element (element in Flash, string in RAM)*
- `#define gslc_ElemCreateTxt_P_R_ext(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, strLength, pFont, colTxt, colTxtGlow, colFrame, colFill, nAlignTxt, nMarginX, nMarginY, bFrameEn, bFillEn, bClickEn, bGlowEn, pfuncXEvent, pfuncXDraw, pfuncXTouch, pfuncXTick)`  
*Create a read-write text element (element in Flash, string in RAM) with extended customization options.*
- `#define gslc_ElemCreateBox_P(pGui, nElemId, nPage, nX, nY, nW, nH, colFrame, colFill, bFrameEn, bFillEn, pfuncXDraw, pfuncXTick)`  
*Create a read-only box element.*
- `#define gslc_ElemCreateLine_P(pGui, nElemId, nPage, nX0, nY0, nX1, nY1, colFill)`  
*Create a read-only line element.*
- `#define gslc_ElemCreateBtnTxt_P(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, pFont, colTxt, colFrame, colFill, colFrameGlow, colFillGlow, nAlignTxt, bFrameEn, bFillEn, callFunc, extraData)`  
*Create a text button element.*

### 7.13.1 Detailed Description

Macros that represent element creation routines based in FLASH memory.

### 7.13.2 Macro Definition Documentation

#### 7.13.2.1 `gslc_ElemCreateBox_P`

```
#define gslc_ElemCreateBox_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    colFrame,  
    colFill,  
    bFrameEn,  
    bFillEn,  
    pfuncXDraw,  
    pfuncXTick )
```

Create a read-only box element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>bFrameEn</i>	True if framed, false otherwise
in	<i>bFillEn</i>	True if filled, false otherwise
in	<i>pfuncXDraw</i>	Pointer to custom draw callback (or NULL if default)
in	<i>pfuncXTick</i>	Pointer to custom tick callback (or NULL if default)

**7.13.2.2 gslc\_ElemCreateBtnTxt\_P**

```
#define gslc_ElemCreateBtnTxt_P(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    strTxt,
    pFont,
    colTxt,
    colFrame,
    colFill,
    colFrameGlow,
    colFillGlow,
    nAlignTxt,
    bFrameEn,
    bFillEn,
    callFunc,
    extraData )
```

Create a text button element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element

## Parameters

in	<i>strTxt</i>	Text string to display
in	<i>pFont</i>	Pointer to font resource
in	<i>colTxt</i>	Color for the text
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>colFrameGlow</i>	Color for the frame when glowing
in	<i>colFillGlow</i>	Color for the fill when glowing
in	<i>nAlignTxt</i>	Text alignment
in	<i>bFrameEn</i>	True if framed, false otherwise
in	<i>bFillEn</i>	True if filled, false otherwise
in	<i>callFunc</i>	Callback function for button press
in	<i>extraData</i>	Ptr to extended data structure

## 7.13.2.3 gslc\_ElemCreateLine\_P

```
#define gslc_ElemCreateLine_P(
    pGui,
    nElemId,
    nPage,
    nX0,
    nY0,
    nX1,
    nY1,
    colFill )
```

Create a read-only line element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX0</i>	X coordinate of line start
in	<i>nY0</i>	Y coordinate of line start
in	<i>nX1</i>	X coordinate of line end
in	<i>nY1</i>	Y coordinate of line end
in	<i>colFill</i>	Color for the line

## 7.13.2.4 gslc\_ElemCreateTxt\_P

```
#define gslc_ElemCreateTxt_P(
    pGui,
```

```

    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    strTxt,
    pFont,
    colTxt,
    colFrame,
    colFill,
    nAlignTxt,
    bFrameEn,
    bFillEn )

```

Create a read-only text element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>strTxt</i>	Text string to display
in	<i>pFont</i>	Pointer to font resource
in	<i>colTxt</i>	Color for the text
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>nAlignTxt</i>	Text alignment
in	<i>bFrameEn</i>	True if framed, false otherwise
in	<i>bFillEn</i>	True if filled, false otherwise

#### 7.13.2.5 gslc\_ElemCreateTxt\_P\_R

```

#define gslc_ElemCreateTxt_P_R(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    strTxt,
    strLength,
    pFont,
    colTxt,
    colFrame,
    colFill,

```

```

    nAlignTxt,
    bFrameEn,
    bFillEn )

```

Create a read-write text element (element in Flash, string in RAM)

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>strTxt</i>	Text string to display
in	<i>strLength</i>	Length of text string
in	<i>pFont</i>	Pointer to font resource
in	<i>colTxt</i>	Color for the text
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>nAlignTxt</i>	Text alignment
in	<i>bFrameEn</i>	True if framed, false otherwise
in	<i>bFillEn</i>	True if filled, false otherwise

#### 7.13.2.6 gslc\_ElemCreateTxt\_P\_R\_ext

```

#define gslc_ElemCreateTxt_P_R_ext (
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    strTxt,
    strLength,
    pFont,
    colTxt,
    colTxtGlow,
    colFrame,
    colFill,
    nAlignTxt,
    nMarginX,
    nMarginY,
    bFrameEn,
    bFillEn,
    bClickEn,
    bGlowEn,
    pfuncXEvent,
    pfuncXDraw,

```

```

    pfuncXTouch,
    pfuncXTick )

```

Create a read-write text element (element in Flash, string in RAM) with extended customization options.

#### Parameters

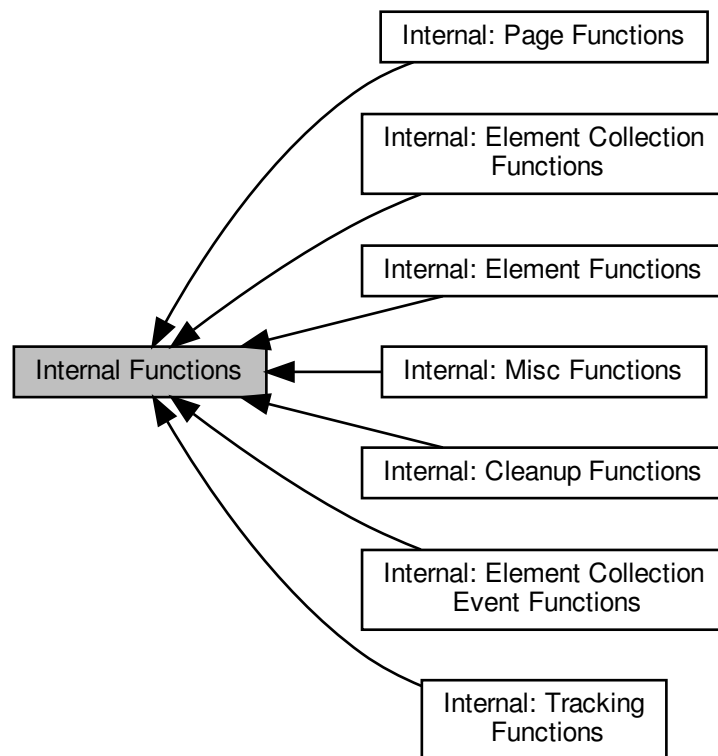
in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>strTxt</i>	Text string to display
in	<i>strLength</i>	Length of text string
in	<i>pFont</i>	Pointer to font resource
in	<i>colTxt</i>	Color for the text
in	<i>colTxtGlow</i>	Color for the text when glowing
in	<i>colFrame</i>	Color for the frame
in	<i>colFill</i>	Color for the fill
in	<i>nAlignTxt</i>	Text alignment
in	<i>nMarginX</i>	Text margin (X offset)
in	<i>nMarginY</i>	Text margin (Y offset)
in	<i>bFrameEn</i>	True if framed, false otherwise
in	<i>bFillEn</i>	True if filled, false otherwise
in	<i>bClickEn</i>	True if accept click events, false otherwise
in	<i>bGlowEn</i>	True if supports glow state, false otherwise
in	<i>pfuncXEvent</i>	Callback function ptr for Event
in	<i>pfuncXDraw</i>	Callback function ptr for Redraw
in	<i>pfuncXTouch</i>	Callback function ptr for Touch
in	<i>pfuncXTick</i>	Callback function ptr for Timer tick



## 7.14 Internal Functions

These functions are internal to the GUIslice implementation and are not intended to be called by user code and subject to change even in minor releases.

Collaboration diagram for Internal Functions:



### Modules

- [Internal: Misc Functions](#)
- [Internal: Element Functions](#)
- [Internal: Page Functions](#)
- [Internal: Element Collection Functions](#)
- [Internal: Element Collection Event Functions](#)
- [Internal: Tracking Functions](#)
- [Internal: Cleanup Functions](#)

### Variables

- `int16_t gslc_tsRect::x`  
*X coordinate of corner.*
- `int16_t gslc_tsRect::y`

- *Y coordinate of corner.*
- `uint16_t gslc_tsRect::w`  
*Width of region.*
- `uint16_t gslc_tsRect::h`  
*Height of region.*
- `int16_t gslc_tsPt::x`  
*X coordinate.*
- `int16_t gslc_tsPt::y`  
*Y coordinate.*
- `uint8_t gslc_tsColor::r`  
*RGB red value.*
- `uint8_t gslc_tsColor::g`  
*RGB green value.*
- `uint8_t gslc_tsColor::b`  
*RGB blue value.*
- `gslc_tsRect gslc_tsRectState::rFocus`
- `gslc_tsRect gslc_tsRectState::rFull`
- `gslc_tsRect gslc_tsRectState::rInner`
- `gslc_tsColor gslc_tsRectState::colFocus`
- `gslc_tsColor gslc_tsRectState::colFrm`
- `gslc_tsColor gslc_tsRectState::colInner`
- `gslc_tsColor gslc_tsRectState::colBack`
- `gslc_tsColor gslc_tsRectState::colTxtFore`
- `gslc_tsColor gslc_tsRectState::colTxtBack`
- `gslc_teEventType gslc_tsEvent::eType`  
*Event type.*
- `uint8_t gslc_tsEvent::nSubType`  
*Event sub-type.*
- `void * gslc_tsEvent::pvScope`  
*Event target scope (eg. Page,Collection,Event)*
- `void * gslc_tsEvent::pvData`  
*Generic data pointer for event.*
- `gslc_teTouch gslc_tsEventTouch::eTouch`  
*Touch state.*
- `int16_t gslc_tsEventTouch::nX`  
*Touch X coordinate (or param1)*
- `int16_t gslc_tsEventTouch::nY`  
*Touch Y coordinate (or param2)*
- `int16_t gslc_tsFont::nId`  
*Font ID specified by user.*
- `gslc_teFontRefType gslc_tsFont::eFontRefType`  
*Font reference type.*
- `gslc_teFontRefMode gslc_tsFont::eFontRefMode`  
*Font reference mode.*
- `const void * gslc_tsFont::pvFont`  
*Void ptr to the font reference (type defined by driver)*
- `uint16_t gslc_tsFont::nSize`  
*Font size.*
- `const unsigned char * gslc_tsImgRef::plmgBuf`  
*Pointer to input image buffer in memory [RAM,FLASH].*
- `const char * gslc_tsImgRef::pFname`

- Pathname to input image file [FILE,SD].*
  - [gslc\\_tslmgRefFlags](#) [gslc\\_tslmgRef::elmgFlags](#)  
*Image reference flags.*
  - `void * gslc\_tslmgRef::pvlmgRaw`  
*Ptr to raw output image data (for pre-loaded images)*
  - [gslc\\_tsElem](#) \* [gslc\\_tsElemRef::pElem](#)  
*Pointer to element in memory [RAM,FLASH].*
  - [gslc\\_teElemRefFlags](#) [gslc\\_tsElemRef::eElemFlags](#)  
*Element reference flags.*
  - `int16_t gslc\_tsElem::nId`  
*Element ID specified by user.*
  - `uint16_t gslc\_tsElem::nFeatures`  
*Element feature vector (appearance/behavior)*
  - `int16_t gslc\_tsElem::nType`  
*Element type enumeration.*
  - [gslc\\_tsRect](#) [gslc\\_tsElem::rElem](#)  
*Rect region containing element.*
  - `int16_t gslc\_tsElem::nGroup`  
*Group ID that the element belongs to.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemFrame](#)  
*Color for frame.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemFill](#)  
*Color for background fill.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemFrameGlow](#)  
*Color to use for frame when glowing.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemFillGlow](#)  
*Color to use for fill when glowing.*
  - [gslc\\_tslmgRef](#) [gslc\\_tsElem::slmgRefNorm](#)  
*Image reference to draw (normal)*
  - [gslc\\_tslmgRef](#) [gslc\\_tsElem::slmgRefGlow](#)  
*Image reference to draw (glowing)*
  - [gslc\\_tsElemRef](#) \* [gslc\\_tsElem::pElemRefParent](#)  
*Parent element reference.*
  - `char * gslc\_tsElem::pStrBuf`  
*Ptr to text string buffer to overlay.*
  - `uint8_t gslc\_tsElem::nStrBufMax`  
*Size of string buffer.*
  - [gslc\\_teTxtFlags](#) [gslc\\_tsElem::eTxtFlags](#)  
*Flags associated with text buffer.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemText](#)  
*Color of overlay text.*
  - [gslc\\_tsColor](#) [gslc\\_tsElem::colElemTextGlow](#)  
*Color of overlay text when glowing.*
  - `int8_t gslc\_tsElem::eTxtAlign`  
*Alignment of overlay text.*
  - `int8_t gslc\_tsElem::nTxtMarginX`  
*Margin of overlay text within rect region (x offset)*
  - `int8_t gslc\_tsElem::nTxtMarginY`  
*Margin of overlay text within rect region (y offset)*
  - [gslc\\_tsFont](#) \* [gslc\\_tsElem::pTxtFont](#)  
*Ptr to Font for overlay text.*

- `void * gslc_tsElem::pXData`  
*Ptr to extended data structure.*
- `GSLC_CB_EVENT gslc_tsElem::pfuncXEvent`  
*UNUSED: Callback func ptr for event tree (draw,touch,tick)*
- `GSLC_CB_DRAW gslc_tsElem::pfuncXDraw`  
*Callback func ptr for custom drawing.*
- `GSLC_CB_TOUCH gslc_tsElem::pfuncXTouch`  
*Callback func ptr for touch.*
- `GSLC_CB_TICK gslc_tsElem::pfuncXTick`  
*Callback func ptr for timer/main loop tick.*
- `gslc_tsElem * gslc_tsCollect::asElem`  
*Array of elements.*
- `uint16_t gslc_tsCollect::nElemMax`  
*Maximum number of elements to allocate (in RAM)*
- `uint16_t gslc_tsCollect::nElemCnt`  
*Number of elements allocated.*
- `int16_t gslc_tsCollect::nElemAutoldNext`  
*Next Element ID for auto-assignment.*
- `gslc_tsElemRef * gslc_tsCollect::asElemRef`  
*Array of element references.*
- `uint16_t gslc_tsCollect::nElemRefMax`  
*Maximum number of element references to allocate.*
- `uint16_t gslc_tsCollect::nElemRefCnt`  
*Number of element references allocated.*
- `gslc_tsElemRef * gslc_tsCollect::pElemRefTracked`  
*Element reference currently being touch-tracked (NULL for none)*
- `int16_t gslc_tsCollect::nElemIndTracked`  
*Element index currently being touch-tracked (GSLC\_IND\_NONE for none)*
- `gslc_tsCollect gslc_tsPage::sCollect`  
*Collection of elements on page.*
- `int16_t gslc_tsPage::nPageId`  
*Page identifier.*
- `gslc_tsRect gslc_tsPage::rBounds`  
*Bounding rect for page elements.*
- `gslc_teInputRawEvent gslc_tsInputMap::eEvent`  
*The input event.*
- `int16_t gslc_tsInputMap::nVal`  
*The value associated with the input event.*
- `gslc_teAction gslc_tsInputMap::eAction`  
*Resulting action.*
- `int16_t gslc_tsInputMap::nActionVal`  
*The value for the output action.*
- `uint16_t gslc_tsGui::nDispW`  
*Width of the display (pixels)*
- `uint16_t gslc_tsGui::nDispH`  
*Height of the display (pixels)*
- `uint16_t gslc_tsGui::nDisp0W`  
*Width of the display (pixels) in native orientation.*
- `uint16_t gslc_tsGui::nDisp0H`  
*Height of the display (pixels) in native orientation.*
- `uint8_t gslc_tsGui::nDispDepth`

- Bit depth of display (bits per pixel)*
- `uint8_t gslc_tsGui::nRotation`
  - Adafruit GFX Rotation of display.*
- `uint8_t gslc_tsGui::nTouchRotation`
  - Touchscreen rotation offset vs display.*
- `uint8_t gslc_tsGui::nSwapXY`
  - Adafruit GFX Touch Swap x and y axes.*
- `uint8_t gslc_tsGui::nFlipX`
  - Adafruit GFX Touch Flip x axis.*
- `uint8_t gslc_tsGui::nFlipY`
  - Adafruit GFX Touch Flip y axis.*
- `int16_t gslc_tsGui::nTouchCalXMin`
  - Calibration X minimum reading.*
- `int16_t gslc_tsGui::nTouchCalXMax`
  - Calibration X maximum reading.*
- `int16_t gslc_tsGui::nTouchCalYMin`
  - Calibration Y minimum reading.*
- `int16_t gslc_tsGui::nTouchCalYMax`
  - Calibration Y maximum reading.*
- `int16_t gslc_tsGui::nTouchCalPressMin`
  - Calibration minimum pressure threshold.*
- `int16_t gslc_tsGui::nTouchCalPressMax`
  - Calibration maximum pressure threshold.*
- `gslc_tsFont * gslc_tsGui::asFont`
  - Collection of loaded fonts.*
- `uint8_t gslc_tsGui::nFontMax`
  - Maximum number of fonts to allocate.*
- `uint8_t gslc_tsGui::nFontCnt`
  - Number of fonts allocated.*
- `uint8_t gslc_tsGui::nRoundRadius`
  - Radius for rounded elements.*
- `gslc_tsColor gslc_tsGui::sTransCol`
  - Color used for transparent image regions (GSLC\_BMP\_TRANS\_EN=1)*
- `gslc_tsElem gslc_tsGui::sElemTmp`
  - Temporary element.*
- `gslc_tsElemRef gslc_tsGui::sElemRefTmp`
  - Temporary element reference.*
- `gslc_tsElem gslc_tsGui::sElemTmpProg`
  - Temporary element for Flash compatibility.*
- `gslc_telnitStat gslc_tsGui::elnitStatTouch`
  - Status of touch initialization.*
- `int16_t gslc_tsGui::nTouchLastX`
  - Last touch event X coord.*
- `int16_t gslc_tsGui::nTouchLastY`
  - Last touch event Y coord.*
- `uint16_t gslc_tsGui::nTouchLastPress`
  - Last touch event pressure (0=none)*
- `bool gslc_tsGui::bTouchRemapEn`
  - Enable touch remapping?*
- `bool gslc_tsGui::bTouchRemapYX`
  - Enable touch controller swapping of X & Y.*

- `bool gslc_tsGui::bTouchEn`  
*Enable reaction to touch events.*
- `void * gslc_tsGui::pvDriver`  
*Driver-specific members (gslc\_tsDriver\*)*
- `bool gslc_tsGui::bRedrawNeeded`  
*Does anything on page require redraw?*
- `bool gslc_tsGui::bRedrawPartialEn`  
*Driver supports partial page redraw.*
- `bool gslc_tsGui::bEventPending`  
*Is there an event pending?*
- `gslc_tsEventTouch gslc_tsGui::sEventTouchPend`  
*A touch event that has been deferred (if bEventPending=true)*
- `gslc_tsEvent gslc_tsGui::sEventPend`  
*An event that has been deferred (if bEventPending=true)*
- `gslc_tsImgRef gslc_tsGui::sImgRefBkgnd`  
*Image reference for background.*
- `uint8_t gslc_tsGui::nFrameRateCnt`  
*Diagnostic frame rate count.*
- `uint8_t gslc_tsGui::nFrameRateStart`  
*Diagnostic frame rate timestamp.*
- `gslc_tsPage * gslc_tsGui::asPage`  
*Array of all pages defined in system.*
- `uint8_t gslc_tsGui::nPageMax`  
*Maximum number of pages that can be defined.*
- `uint8_t gslc_tsGui::nPageCnt`  
*Current number of pages defined.*
- `gslc_tsPage * gslc_tsGui::apPageStack [GSLC_STACK__MAX]`  
*Stack of pages.*
- `bool gslc_tsGui::abPageStackActive [GSLC_STACK__MAX]`  
*Whether page in stack can receive touch events.*
- `bool gslc_tsGui::abPageStackDoDraw [GSLC_STACK__MAX]`  
*Whether page in stack is still actively drawn.*
- `bool gslc_tsGui::bScreenNeedRedraw`  
*Screen requires a redraw.*
- `bool gslc_tsGui::bScreenNeedFlip`  
*Screen requires a page flip.*
- `bool gslc_tsGui::bInvalidateEn`  
*A region of the display has been invalidated.*
- `gslc_tsRect gslc_tsGui::rInvalidateRect`  
*The rect region that has been invalidated.*
- `GSLC_CB_PIN_POLL gslc_tsGui::pfuncPinPoll`  
*Callback func ptr for pin polling.*
- `gslc_tsInputMap * gslc_tsGui::asInputMap`  
*Array of input maps.*
- `uint8_t gslc_tsGui::nInputMapMax`  
*Maximum number of input maps.*
- `uint8_t gslc_tsGui::nInputMapCnt`  
*Current number of input maps.*
- `uint8_t gslc_tsGui::nInputMode`  
*Input mode: 0=navigate, 1=edit.*
- `int16_t gslc_tsGui::nFocusPageInd`

- *Index of page in stack currently in focus.*  
`gslc_tsPage * gslc_tsGui::pFocusPage`
- *Page ptr currently in focus.*  
`gslc_tsElemRef * gslc_tsGui::pFocusElemRef`
- *Reference to element in focus.*  
`int16_t gslc_tsGui::nFocusElemInd`
- *Index of element in page currently in focus.*  
`int16_t gslc_tsGui::nFocusElemMax`
- *Max number of elements in page in focus.*  
`gslc_tsColor gslc_tsGui::colFocusNone`
- *Focus frame color when not in focus (typically background color)*  
`gslc_tsColor gslc_tsGui::colFocus`
- *Focus frame color when in focus.*  
`gslc_tsColor gslc_tsGui::colFocusEdit`
- *Focus frame color when in focus and edit mode.*  
`int16_t gslc_tsGui::nFocusSavedPageInd`
- *Focus page index saved prior to popup/overlay.*  
`int16_t gslc_tsGui::nFocusSavedElemInd`
- *Focus element index saved prior to popup/overlay.*

### 7.14.1 Detailed Description

These functions are internal to the GUIslice implementation and are not intended to be called by user code and subject to change even in minor releases.

- The following functions are generally not required for typical users of GUIslice. However, for advanced usage more direct access may be required.

### 7.14.2 Variable Documentation

#### 7.14.2.1 abPageStackActive

```
bool gslc_tsGui::abPageStackActive[GSLC_STACK__MAX]
```

Whether page in stack can receive touch events.

#### 7.14.2.2 abPageStackDoDraw

```
bool gslc_tsGui::abPageStackDoDraw[GSLC_STACK__MAX]
```

Whether page in stack is still actively drawn.

#### 7.14.2.3 apPageStack

```
gslc_tsPage* gslc_tsGui::apPageStack[GSLC_STACK__MAX]
```

Stack of pages.

#### 7.14.2.4 asElem

```
gslc_tsElem* gslc_tsCollect::asElem
```

Array of elements.

#### 7.14.2.5 asElemRef

```
gslc_tsElemRef* gslc_tsCollect::asElemRef
```

Array of element references.

#### 7.14.2.6 asFont

```
gslc_tsFont* gslc_tsGui::asFont
```

Collection of loaded fonts.

#### 7.14.2.7 asInputMap

```
gslc_tsInputMap* gslc_tsGui::asInputMap
```

Array of input maps.

#### 7.14.2.8 asPage

```
gslc_tsPage* gslc_tsGui::asPage
```

Array of all pages defined in system.



#### 7.14.2.9 b

```
uint8_t gslc_tsColor::b
```

RGB blue value.

#### 7.14.2.10 bEventPending

```
bool gslc_tsGui::bEventPending
```

Is there an event pending?

#### 7.14.2.11 bInvalidateEn

```
bool gslc_tsGui::bInvalidateEn
```

A region of the display has been invalidated.

#### 7.14.2.12 bRedrawNeeded

```
bool gslc_tsGui::bRedrawNeeded
```

Does anything on page require redraw?

#### 7.14.2.13 bRedrawPartialEn

```
bool gslc_tsGui::bRedrawPartialEn
```

Driver supports partial page redraw.

If true, only changed elements are redrawn during next page redraw command. If false, entire page is redrawn when any element has been updated prior to next page redraw command.

#### 7.14.2.14 bScreenNeedFlip

```
bool gslc_tsGui::bScreenNeedFlip
```

Screen requires a page flip.

#### 7.14.2.15 bScreenNeedRedraw

```
bool gslc_tsGui::bScreenNeedRedraw
```

Screen requires a redraw.

#### 7.14.2.16 bTouchEn

```
bool gslc_tsGui::bTouchEn
```

Enable reaction to touch events.

#### 7.14.2.17 bTouchRemapEn

```
bool gslc_tsGui::bTouchRemapEn
```

Enable touch remapping?

#### 7.14.2.18 bTouchRemapYX

```
bool gslc_tsGui::bTouchRemapYX
```

Enable touch controller swapping of X & Y.

#### 7.14.2.19 colBack

```
gslc_tsColor gslc_tsRectState::colBack
```

#### 7.14.2.20 colElemFill

```
gslc_tsColor gslc_tsElem::colElemFill
```

Color for background fill.

**7.14.2.21 colElemFillGlow**

```
gslc_tsColor gslc_tsElem::colElemFillGlow
```

Color to use for fill when glowing.

**7.14.2.22 colElemFrame**

```
gslc_tsColor gslc_tsElem::colElemFrame
```

Color for frame.

**7.14.2.23 colElemFrameGlow**

```
gslc_tsColor gslc_tsElem::colElemFrameGlow
```

Color to use for frame when glowing.

**7.14.2.24 colElemText**

```
gslc_tsColor gslc_tsElem::colElemText
```

Color of overlay text.

**7.14.2.25 colElemTextGlow**

```
gslc_tsColor gslc_tsElem::colElemTextGlow
```

Color of overlay text when glowing.

**7.14.2.26 colFocus** [1/2]

```
gslc_tsColor gslc_tsRectState::colFocus
```

#### 7.14.2.27 colFocus [2/2]

```
gslc_tsColor gslc_tsGui::colFocus
```

Focus frame color when in focus.

#### 7.14.2.28 colFocusEdit

```
gslc_tsColor gslc_tsGui::colFocusEdit
```

Focus frame color when in focus and edit mode.

#### 7.14.2.29 colFocusNone

```
gslc_tsColor gslc_tsGui::colFocusNone
```

Focus frame color when not in focus (typically background color)

#### 7.14.2.30 colFrm

```
gslc_tsColor gslc_tsRectState::colFrm
```

#### 7.14.2.31 colInner

```
gslc_tsColor gslc_tsRectState::colInner
```

#### 7.14.2.32 colTxtBack

```
gslc_tsColor gslc_tsRectState::colTxtBack
```

#### 7.14.2.33 colTxtFore

```
gslc_tsColor gslc_tsRectState::colTxtFore
```

#### 7.14.2.34 eAction

`gslc_teAction` `gslc_tsInputMap::eAction`

Resulting action.

#### 7.14.2.35 eElemFlags

`gslc_teElemRefFlags` `gslc_tsElemRef::eElemFlags`

Element reference flags.

#### 7.14.2.36 eEvent

`gslc_teInputRawEvent` `gslc_tsInputMap::eEvent`

The input event.

#### 7.14.2.37 eFontRefMode

`gslc_teFontRefMode` `gslc_tsFont::eFontRefMode`

Font reference mode.

#### 7.14.2.38 eFontRefType

`gslc_teFontRefType` `gslc_tsFont::eFontRefType`

Font reference type.

#### 7.14.2.39 eImgFlags

`gslc_teImgRefFlags` `gslc_tsImgRef::eImgFlags`

Image reference flags.

#### 7.14.2.40 eInitStatTouch

`gslc_teInitStat` `gslc_tsGui::eInitStatTouch`

Status of touch initialization.

#### 7.14.2.41 eTouch

`gslc_teTouch` `gslc_tsEventTouch::eTouch`

Touch state.

#### 7.14.2.42 eTxtAlign

`int8_t` `gslc_tsElem::eTxtAlign`

Alignment of overlay text.

#### 7.14.2.43 eTxtFlags

`gslc_teTxtFlags` `gslc_tsElem::eTxtFlags`

Flags associated with text buffer.

#### 7.14.2.44 eType

`gslc_teEventType` `gslc_tsEvent::eType`

Event type.

#### 7.14.2.45 g

`uint8_t` `gslc_tsColor::g`

RGB green value.

**7.14.2.46 h**

```
uint16_t gslc_tsRect::h
```

Height of region.

**7.14.2.47 nActionVal**

```
int16_t gslc_tsInputMap::nActionVal
```

The value for the output action.

**7.14.2.48 nDisp0H**

```
uint16_t gslc_tsGui::nDisp0H
```

Height of the display (pixels) in native orientation.

**7.14.2.49 nDisp0W**

```
uint16_t gslc_tsGui::nDisp0W
```

Width of the display (pixels) in native orientation.

**7.14.2.50 nDispDepth**

```
uint8_t gslc_tsGui::nDispDepth
```

Bit depth of display (bits per pixel)

**7.14.2.51 nDispH**

```
uint16_t gslc_tsGui::nDispH
```

Height of the display (pixels)

#### 7.14.2.52 nDispW

```
uint16_t gslc_tsGui::nDispW
```

Width of the display (pixels)

#### 7.14.2.53 nElemAutoIdNext

```
int16_t gslc_tsCollect::nElemAutoIdNext
```

Next Element ID for auto-assignment.

#### 7.14.2.54 nElemCnt

```
uint16_t gslc_tsCollect::nElemCnt
```

Number of elements allocated.

#### 7.14.2.55 nElemIndTracked

```
int16_t gslc_tsCollect::nElemIndTracked
```

Element index currently being touch-tracked (GSLC\_IND\_NONE for none)

#### 7.14.2.56 nElemMax

```
uint16_t gslc_tsCollect::nElemMax
```

Maximum number of elements to allocate (in RAM)

#### 7.14.2.57 nElemRefCnt

```
uint16_t gslc_tsCollect::nElemRefCnt
```

Number of element references allocated.



**7.14.2.58 nElemRefMax**

```
uint16_t gslc_tsCollect::nElemRefMax
```

Maximum number of element references to allocate.

**7.14.2.59 nFeatures**

```
uint16_t gslc_tsElem::nFeatures
```

Element feature vector (appearance/behavior)

**7.14.2.60 nFlipX**

```
uint8_t gslc_tsGui::nFlipX
```

Adafruit GFX Touch Flip x axis.

**7.14.2.61 nFlipY**

```
uint8_t gslc_tsGui::nFlipY
```

Adafruit GFX Touch Flip y axis.

**7.14.2.62 nFocusElemInd**

```
int16_t gslc_tsGui::nFocusElemInd
```

Index of element in page currently in focus.

**7.14.2.63 nFocusElemMax**

```
int16_t gslc_tsGui::nFocusElemMax
```

Max number of elements in page in focus.

**7.14.2.64 nFocusPageInd**

```
int16_t gslc_tsGui::nFocusPageInd
```

Index of page in stack currently in focus.

**7.14.2.65 nFocusSavedElemInd**

```
int16_t gslc_tsGui::nFocusSavedElemInd
```

Focus element index saved prior to popup/overlay.

**7.14.2.66 nFocusSavedPageInd**

```
int16_t gslc_tsGui::nFocusSavedPageInd
```

Focus page index saved prior to popup/overlay.

**7.14.2.67 nFontCnt**

```
uint8_t gslc_tsGui::nFontCnt
```

Number of fonts allocated.

**7.14.2.68 nFontMax**

```
uint8_t gslc_tsGui::nFontMax
```

Maximum number of fonts to allocate.

**7.14.2.69 nFrameRateCnt**

```
uint8_t gslc_tsGui::nFrameRateCnt
```

Diagnostic frame rate count.

**7.14.2.70 nFrameRateStart**

```
uint8_t gslc_tsGui::nFrameRateStart
```

Diagnostic frame rate timestamp.

**7.14.2.71 nGroup**

```
int16_t gslc_tsElem::nGroup
```

Group ID that the element belongs to.

**7.14.2.72 nId [1/2]**

```
int16_t gslc_tsFont::nId
```

Font ID specified by user.

**7.14.2.73 nId [2/2]**

```
int16_t gslc_tsElem::nId
```

Element ID specified by user.

**7.14.2.74 nInputMapCnt**

```
uint8_t gslc_tsGui::nInputMapCnt
```

Current number of input maps.

**7.14.2.75 nInputMapMax**

```
uint8_t gslc_tsGui::nInputMapMax
```

Maximum number of input maps.

#### 7.14.2.76 nInputMode

```
uint8_t gslc_tsGui::nInputMode
```

Input mode: 0=navigate, 1=edit.

#### 7.14.2.77 nPageCnt

```
uint8_t gslc_tsGui::nPageCnt
```

Current number of pages defined.

#### 7.14.2.78 nPageId

```
int16_t gslc_tsPage::nPageId
```

Page identifier.

#### 7.14.2.79 nPageMax

```
uint8_t gslc_tsGui::nPageMax
```

Maximum number of pages that can be defined.

#### 7.14.2.80 nRotation

```
uint8_t gslc_tsGui::nRotation
```

Adafruit GFX Rotation of display.

#### 7.14.2.81 nRoundRadius

```
uint8_t gslc_tsGui::nRoundRadius
```

Radius for rounded elements.

**7.14.2.82 nSize**

```
uint16_t gslc_tsFont::nSize
```

Font size.

**7.14.2.83 nStrBufMax**

```
uint8_t gslc_tsElem::nStrBufMax
```

Size of string buffer.

**7.14.2.84 nSubType**

```
uint8_t gslc_tsEvent::nSubType
```

Event sub-type.

**7.14.2.85 nSwapXY**

```
uint8_t gslc_tsGui::nSwapXY
```

Adafruit GFX Touch Swap x and y axes.

**7.14.2.86 nTouchCalPressMax**

```
int16_t gslc_tsGui::nTouchCalPressMax
```

Calibration maximum pressure threshold.

**7.14.2.87 nTouchCalPressMin**

```
int16_t gslc_tsGui::nTouchCalPressMin
```

Calibration minimum pressure threshold.

**7.14.2.88 nTouchCalXMax**

```
int16_t gslc_tsGui::nTouchCalXMax
```

Calibration X maximum reading.

**7.14.2.89 nTouchCalXMin**

```
int16_t gslc_tsGui::nTouchCalXMin
```

Calibration X minimum reading.

**7.14.2.90 nTouchCalYMax**

```
int16_t gslc_tsGui::nTouchCalYMax
```

Calibration Y maximum reading.

**7.14.2.91 nTouchCalYMin**

```
int16_t gslc_tsGui::nTouchCalYMin
```

Calibration Y minimum reading.

**7.14.2.92 nTouchLastPress**

```
uint16_t gslc_tsGui::nTouchLastPress
```

Last touch event pressure (0=none))

**7.14.2.93 nTouchLastX**

```
int16_t gslc_tsGui::nTouchLastX
```

Last touch event X coord.

**7.14.2.94 nTouchLastY**

```
int16_t gslc_tsGui::nTouchLastY
```

Last touch event Y coord.

**7.14.2.95 nTouchRotation**

```
uint8_t gslc_tsGui::nTouchRotation
```

Touchscreen rotation offset vs display.

**7.14.2.96 nTxtMarginX**

```
int8_t gslc_tsElem::nTxtMarginX
```

Margin of overlay text within rect region (x offset)

**7.14.2.97 nTxtMarginY**

```
int8_t gslc_tsElem::nTxtMarginY
```

Margin of overlay text within rect region (y offset)

**7.14.2.98 nType**

```
int16_t gslc_tsElem::nType
```

Element type enumeration.

**7.14.2.99 nVal**

```
int16_t gslc_tsInputMap::nVal
```

The value associated with the input event.

**7.14.2.100 nX**

```
int16_t gslc_tsEventTouch::nX
```

Touch X coordinate (or param1)

**7.14.2.101 nY**

```
int16_t gslc_tsEventTouch::nY
```

Touch Y coordinate (or param2)

**7.14.2.102 pElem**

```
gslc_tsElem* gslc_tsElemRef::pElem
```

Pointer to element in memory [RAM,FLASH].

**7.14.2.103 pElemRefParent**

```
gslc_tsElemRef* gslc_tsElem::pElemRefParent
```

Parent element reference.

Used during redraw to notify parent elements that they require redraw as well. Primary usage is in compound elements. NOTE: Although this field is only used in GLSC\_COMPOUND mode, it is not wrapped in an ifdef because the ElemCreate\*\_P() function macros currently initialize this field.

**7.14.2.104 pElemRefTracked**

```
gslc_tsElemRef* gslc_tsCollect::pElemRefTracked
```

Element reference currently being touch-tracked (NULL for none)

**7.14.2.105 pFname**

```
const char* gslc_tsImgRef::pFname
```

Pathname to input image file [FILE,SD].



**7.14.2.106 pFocusElemRef**

```
gslc_tsElemRef* gslc_tsGui::pFocusElemRef
```

Reference to element in focus.

**7.14.2.107 pFocusPage**

```
gslc_tsPage* gslc_tsGui::pFocusPage
```

Page ptr currently in focus.

**7.14.2.108 pfuncPinPoll**

```
GSLC_CB_PIN_POLL gslc_tsGui::pfuncPinPoll
```

Callback func ptr for pin polling.

**7.14.2.109 pfuncXDraw**

```
GSLC_CB_DRAW gslc_tsElem::pfuncXDraw
```

Callback func ptr for custom drawing.

**7.14.2.110 pfuncXEvent**

```
GSLC_CB_EVENT gslc_tsElem::pfuncXEvent
```

UNUSED: Callback func ptr for event tree (draw,touch,tick)

**7.14.2.111 pfuncXTick**

```
GSLC_CB_TICK gslc_tsElem::pfuncXTick
```

Callback func ptr for timer/main loop tick.

#### 7.14.2.112 pfuncXTouch

```
gslc_cb_touch gslc_tsElem::pfuncXTouch
```

Callback func ptr for touch.

#### 7.14.2.113 pImgBuf

```
const unsigned char* gslc_tsImgRef::pImgBuf
```

Pointer to input image buffer in memory [RAM,FLASH].

#### 7.14.2.114 pStrBuf

```
char* gslc_tsElem::pStrBuf
```

Ptr to text string buffer to overlay.

#### 7.14.2.115 pTxtFont

```
gslc_tsFont* gslc_tsElem::pTxtFont
```

Ptr to Font for overlay text.

#### 7.14.2.116 pvData

```
void* gslc_tsEvent::pvData
```

Generic data pointer for event.

This member is used to either pass a pointer to a simple data datatype (such as Element or Collection) or to a another structure that contains multiple fields.

#### 7.14.2.117 pvDriver

```
void* gslc_tsGui::pvDriver
```

Driver-specific members (gslc\_tsDriver\*)

**7.14.2.118 pvFont**

```
const void* gslc_tsFont::pvFont
```

Void ptr to the font reference (type defined by driver)

**7.14.2.119 pvImgRaw**

```
void* gslc_tsImgRef::pvImgRaw
```

Ptr to raw output image data (for pre-loaded images)

**7.14.2.120 pvScope**

```
void* gslc_tsEvent::pvScope
```

Event target scope (eg. Page,Collection,Event)

**7.14.2.121 pXData**

```
void* gslc_tsElem::pXData
```

Ptr to extended data structure.

**7.14.2.122 r**

```
uint8_t gslc_tsColor::r
```

RGB red value.

**7.14.2.123 rBounds**

```
gslc_tsRect gslc_tsPage::rBounds
```

Bounding rect for page elements.

#### 7.14.2.124 rElem

`gslc_tsRect` `gslc_tsElem::rElem`

Rect region containing element.

#### 7.14.2.125 rFocus

`gslc_tsRect` `gslc_tsRectState::rFocus`

#### 7.14.2.126 rFull

`gslc_tsRect` `gslc_tsRectState::rFull`

#### 7.14.2.127 rInner

`gslc_tsRect` `gslc_tsRectState::rInner`

#### 7.14.2.128 rInvalidateRect

`gslc_tsRect` `gslc_tsGui::rInvalidateRect`

The rect region that has been invalidated.

#### 7.14.2.129 sCollect

`gslc_tsCollect` `gslc_tsPage::sCollect`

Collection of elements on page.

#### 7.14.2.130 sElemRefTmp

`gslc_tsElemRef` `gslc_tsGui::sElemRefTmp`

Temporary element reference.

**7.14.2.131 sElemTmp**

```
gslc_tsElem gslc_tsGui::sElemTmp
```

Temporary element.

**7.14.2.132 sElemTmpProg**

```
gslc_tsElem gslc_tsGui::sElemTmpProg
```

Temporary element for Flash compatibility.

**7.14.2.133 sEventPend**

```
gslc_tsEvent gslc_tsGui::sEventPend
```

An event that has been deferred (if bEventPending=true)

**7.14.2.134 sEventTouchPend**

```
gslc_tsEventTouch gslc_tsGui::sEventTouchPend
```

A touch event that has been deferred (if bEventPending=true)

**7.14.2.135 sImgRefBkgnd**

```
gslc_tsImgRef gslc_tsGui::sImgRefBkgnd
```

Image reference for background.

**7.14.2.136 sImgRefGlow**

```
gslc_tsImgRef gslc_tsElem::sImgRefGlow
```

Image reference to draw (glowing)

**7.14.2.137 sImgRefNorm**

```
gslc_tsImgRef gslc_tsElem::sImgRefNorm
```

Image reference to draw (normal)

**7.14.2.138 sTransCol**

```
gslc_tsColor gslc_tsGui::sTransCol
```

Color used for transparent image regions (GSLC\_BMP\_TRANS\_EN=1)

**7.14.2.139 w**

```
uint16_t gslc_tsRect::w
```

Width of region.

**7.14.2.140 x [1/2]**

```
int16_t gslc_tsRect::x
```

X coordinate of corner.

**7.14.2.141 x [2/2]**

```
int16_t gslc_tsPt::x
```

X coordinate.

**7.14.2.142 y [1/2]**

```
int16_t gslc_tsRect::y
```

Y coordinate of corner.

**7.14.2.143 y [2/2]**

```
int16_t gslc_tsPt::y
```

Y coordinate.

## 7.15 Internal: Misc Functions

Collaboration diagram for Internal: Misc Functions:



### Functions

- `gslc_tsImgRef gslc_ResetImage ()`  
*Create a blank image reference structure.*

#### 7.15.1 Detailed Description

#### 7.15.2 Function Documentation

##### 7.15.2.1 `gslc_ResetImage()`

`gslc_tsImgRef` `gslc_ResetImage ( )`

Create a blank image reference structure.

#### Returns

Image reference struct

## 7.16 Internal: Element Functions

Collaboration diagram for Internal: Element Functions:



### Functions

- [gslc\\_tsElem](#) [gslc\\_ElemCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPageId, int16\_t nType, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create a new element with default styling.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsElemRefFlags](#) eFlags)  
*Add the Element to the list of generated elements in the GUI environment.*
- uint8\_t [gslc\\_GetElemRefFlag](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nFlagMask)  
*Get the flags associated with an element reference.*
- void [gslc\\_SetElemRefFlag](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nFlagMask, uint8\_t nFlagVal)  
*Set the flags associated with an element reference.*
- [gslc\\_tsElem](#) \* [gslc\\_GetElemFromRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Returns a pointer to an element from an element reference, copying from FLASH to RAM if element is stored in PROGMEM.*
- [gslc\\_tsElem](#) \* [gslc\\_GetElemFromRefD](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nLineNum)  
*Returns a pointer to an element from an element reference.*
- void \* [gslc\\_GetXDataFromRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nType, int16\_t nLineNum)  
*Returns a pointer to the data structure associated with an extended element.*
- void [gslc\\_ElemSetImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsImgRef](#) sImgRef, [gslc\\_tsImgRefSel](#))  
*Set an element to use a bitmap image.*
- bool [gslc\\_ElemDrawByRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRedrawType](#) eRedraw)  
*Draw an element to the active display.*
- void [gslc\\_ElemDraw](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, int16\_t nElemId)  
*Draw an element to the active display.*
- void [gslc\\_DrawTxtBase](#) ([gslc\\_tsGui](#) \*pGui, char \*pStrBuf, [gslc\\_tsRect](#) rTxt, [gslc\\_tsFont](#) \*pTxtFont, [gslc\\_tsTxtFlags](#) eTxtFlags, int8\_t eTxtAlign, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg, int16\_t nMarginW, int16\_t nMarginH)  
*Draw text with full text justification.*
- void [gslc\\_SetRoundRadius](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRadius)  
*Set the global rounded radius.*
- void [gslc\\_SetFocusCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) colFocusNone, [gslc\\_tsColor](#) colFocus, [gslc\\_tsColor](#) colFocusEdit)  
*Set the global focus color choices.*



### 7.16.1 Detailed Description

### 7.16.2 Function Documentation

#### 7.16.2.1 gslc\_DrawTxtBase()

```
void gslc_DrawTxtBase (
    gslc_tsGui * pGui,
    char * pStrBuf,
    gslc_tsRect rTxt,
    gslc_tsFont * pTxtFont,
    gslc_teTxtFlags eTxtFlags,
    int8_t eTxtAlign,
    gslc_tsColor colTxt,
    gslc_tsColor colBg,
    int16_t nMarginW,
    int16_t nMarginH )
```

Draw text with full text justification.

- This function is usually only required by internal GUIslice rendering operations but is made available for custom element usage as well

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pStrBuf</i>	Pointer to text string buffer
in	<i>rTxt</i>	Rectangle region to contain the text
in	<i>pTxtFont</i>	Pointer to the font
in	<i>eTxtFlags</i>	Text string attributes
in	<i>eTxtAlign</i>	Text alignment / justification mode
in	<i>colTxt</i>	Text foreground color
in	<i>colBg</i>	Text background color
in	<i>nMarginW</i>	Horizontal margin within rect region to keep text away
in	<i>nMarginH</i>	Vertical margin within rect region to keep text away

#### Returns

none

#### 7.16.2.2 gslc\_ElemAdd()

```
gslc_tsElemRef* gslc_ElemAdd (
    gslc_tsGui * pGui,
```

```

    int16_t nPageId,
    gslc_tsElem * pElem,
    gslc_teElemRefFlags eFlags )

```

Add the Element to the list of generated elements in the GUI environment.

- NOTE: The content of pElem is copied so the pointer can be released after the call.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nPageId</i>	Page ID to add element to (GSLC_PAGE_NONE to skip in case of temporary creation for compound elements)
in	<i>pElem</i>	Pointer to Element to add
in	<i>eFlags</i>	Flags describing the element (eg. whether the element should be stored in internal RAM array or is located in Flash/PROGMEM).

#### Returns

Pointer to Element reference or NULL if fail

#### 7.16.2.3 gslc\_ElemCreate()

```

gslc_tsElem gslc_ElemCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPageId,
    int16_t nType,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )

```

Create a new element with default styling.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	User-supplied ID for referencing this element (or GSLC_ID_AUTO to auto-generate)
in	<i>nPageId</i>	The page ID on which this page should be associated
in	<i>nType</i>	Enumeration that indicates the type of element that is requested for creation. The type adjusts the visual representation and default styling.
in	<i>rElem</i>	Rectangle region framing the element
in	<i>pStrBuf</i>	String to copy into element
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf). Only applicable if GSLC_LOCAL_STR=0. Ignored if GSLC_LOCAL_STR=1.)
in	<i>nFontId</i>	Font ID for textual elements

**Returns**

Initialized structure

**7.16.2.4 gslc\_ElemDraw()**

```
void gslc_ElemDraw (
    gslc_tsGui * pGui,
    int16_t nPageId,
    int16_t nElemId )
```

Draw an element to the active display.

- Element is referenced by a page ID and element ID
- Provides similar functionality as ElemDrawByRef() but accepts page and element IDs

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n</i> ↔ <i>PageId</i>	ID of page containing element
in	<i>n</i> ↔ <i>ElemId</i>	ID of element

**Returns**

none

**7.16.2.5 gslc\_ElemDrawByRef()**

```
bool gslc_ElemDrawByRef (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Draw an element to the active display.

- Element is referenced by an element pointer

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element reference to draw
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**7.16.2.6 gslc\_ElemSetImage()**

```
void gslc_ElemSetImage (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsImgRef sImgRef,
    gslc_tsImgRef sImgRefSel )
```

Set an element to use a bitmap image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference to update
in	<i>sImgRef</i>	Image reference (normal state)
in	<i>sImgRefSel</i>	Image reference (glowing state)

**Returns**

none

**7.16.2.7 gslc\_GetElemFromRef()**

```
gslc_tsElem* gslc_GetElemFromRef (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Returns a pointer to an element from an element reference, copying from FLASH to RAM if element is stored in PROGMEM.

This function enables all APIs to work with Elements irrespective of whether they were created in RAM or Flash.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element Reference

**Returns**

Pointer to Element after ensuring that it is accessible from RAM

## 7.16.2.8 gslc\_GetElemFromRefD()

```
gslc_tsElem* gslc_GetElemFromRefD (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nLineNum )
```

Returns a pointer to an element from an element reference.

This is a wrapper for GetElemFromRef() including debug checking for invalid pointers.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element Reference
in	<i>nLineNum</i>	Line number from calling function (ie. <b>LINE</b> )

## Returns

Pointer to Element after ensuring that it is accessible from RAM

## 7.16.2.9 gslc\_GetElemRefFlag()

```
uint8_t gslc_GetElemRefFlag (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nFlagMask )
```

Get the flags associated with an element reference.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Element reference pointer
in	<i>nFlagMask</i>	Flags to read

## Returns

Values associated with the element reference flags (subject to the flag mask)

## 7.16.2.10 gslc\_GetXDataFromRef()

```
void* gslc_GetXDataFromRef (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
```

```

    int16_t nType,
    int16_t nLineNum )

```

Returns a pointer to the data structure associated with an extended element.

- Example usage: `gslc_tsXListbox* pListbox = (gslc_tsXListbox*)gslc_GetXDataFromRef(pGui, pElemRef, GSLC_TYPEX_LISTBOX, LINE);`

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element Reference
in	<i>nType</i>	Expected type indicator (ie. <code>GSLC_TYPEX_*</code> )
in	<i>nLineNum</i>	Line number from calling function (ie. <b>LINE</b> )

#### Returns

Void pointer to extended data (pXData), or NULL if error. Needs to be typecasted accordingly.

#### 7.16.2.11 `gslc_SetElemRefFlag()`

```

void gslc_SetElemRefFlag (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nFlagMask,
    uint8_t nFlagVal )

```

Set the flags associated with an element reference.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Element reference pointer
in	<i>nFlagMask</i>	Flags to read
in	<i>nFlagVal</i>	Values to assign to masked flags

#### Returns

none

#### 7.16.2.12 `gslc_SetFocusCol()`

```

void gslc_SetFocusCol (
    gslc_tsGui * pGui,

```

```

gslc_tsColor colFocusNone,
gslc_tsColor colFocus,
gslc_tsColor colFocusEdit )

```

Set the global focus color choices.

- These colors will be used when depicting focus frames around elements

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>colFocusNone</i>	The color to use when the element is not in focus. This is typically set to match the background color.
in	<i>colFocus</i>	The color to use when the element is in focus.
in	<i>colFocusEdit</i>	The color to use when the element is in edit mode.

#### Returns

none

#### 7.16.2.13 gslc\_SetRoundRadius()

```

void gslc_SetRoundRadius (
    gslc_tsGui * pGui,
    uint8_t nRadius )

```

Set the global rounded radius.

- Used for rounded rectangles

#### Parameters

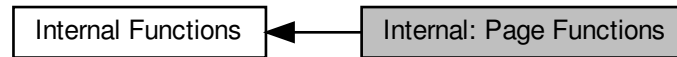
in	<i>pGui</i>	Pointer to GUI
in	<i>nRadius</i>	Radius for rounded elements

#### Returns

none

## 7.17 Internal: Page Functions

Collaboration diagram for Internal: Page Functions:



### Functions

- bool [gslc\\_PageEvent](#) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)  
*Common event handler function for a page.*
- void [gslc\\_PageRedrawGo](#) ([gslc\\_tsGui](#) \*pGui)  
*Redraw all elements on the active page.*
- void [gslc\\_PageFlipSet](#) ([gslc\\_tsGui](#) \*pGui, bool bNeeded)  
*Indicate whether the screen requires page flip.*
- bool [gslc\\_PageFlipGet](#) ([gslc\\_tsGui](#) \*pGui)  
*Get state of pending page flip state.*
- void [gslc\\_PageFlipGo](#) ([gslc\\_tsGui](#) \*pGui)  
*Update the visible screen if page has been marked for flipping.*
- [gslc\\_tsPage](#) \* [gslc\\_PageFindById](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId)  
*Find a page in the GUI by its ID.*
- void [gslc\\_PageRedrawCalc](#) ([gslc\\_tsGui](#) \*pGui)  
*Perform a redraw calculation on the page to determine if additional elements should also be redrawn.*
- [gslc\\_tsEvent](#) [gslc\\_EventCreate](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_teEventType](#) eType, uint8\_t nSubType, void \*pv↔Scope, void \*pvData)  
*Create an event structure.*
- bool [gslc\\_ElemEvent](#) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)  
*Common event handler function for an element.*
- bool [gslc\\_ElemSendEventTouch](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRefTracked, [gslc\\_teTouch](#) e↔Touch, int16\_t nX, int16\_t nY)  
*Trigger an element's touch event.*

### 7.17.1 Detailed Description

### 7.17.2 Function Documentation

#### 7.17.2.1 [gslc\\_ElemEvent\(\)](#)

```
bool gslc_ElemEvent (
    void * pvGui,
    gslc\_tsEvent sEvent )
```

Common event handler function for an element.



**Parameters**

in	<i>pGui</i>	Void pointer to GUI
in	<i>sEvent</i>	Event data structure

**Returns**

true if success, false if fail

**7.17.2.2 gslc\_ElemSendEventTouch()**

```
bool gslc_ElemSendEventTouch (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRefTracked,
    gslc_teTouch eTouch,
    int16_t nX,
    int16_t nY )
```

Trigger an element's touch event.

This is an optional behavior useful in some extended element types.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRefTracked</i>	Pointer to tracked Element reference (or NULL for none))
in	<i>eTouch</i>	Touch event type
in	<i>nX</i>	X coordinate of event (absolute coordinate)
in	<i>nY</i>	Y coordinate of event (absolute coordinate)

**Returns**

true if success, false if error

**7.17.2.3 gslc\_EventCreate()**

```
gslc_tsEvent gslc_EventCreate (
    gslc_tsGui * pGui,
    gslc_teEventType eType,
    uint8_t nSubType,
    void * pvScope,
    void * pvData )
```

Create an event structure.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>eType</i>	Event type (draw, touch, tick, etc.)
in	<i>nSubType</i>	Refinement of event type (or 0 if unused)
in	<i>pvScope</i>	Void ptr to object receiving event so that the event handler will have the context
in	<i>pvData</i>	Void ptr to additional data associated with the event (eg. coordinates for touch events)

**Returns**

None

**7.17.2.4 gslc\_PageEvent()**

```
bool gslc_PageEvent (
    void * pvGui,
    gslc_tsEvent sEvent )
```

Common event handler function for a page.

**Parameters**

in	<i>pvGui</i>	Void pointer to GUI
in	<i>sEvent</i>	Event data structure

**Returns**

true if success, false if fail

**7.17.2.5 gslc\_PageFindById()**

```
gslc_tsPage* gslc_PageFindById (
    gslc_tsGui * pGui,
    int16_t nPageId )
```

Find a page in the GUI by its ID.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ PageId</i>	Page ID to search

**Returns**

Ptr to a page or NULL if none found

**7.17.2.6 gslc\_PageFlipGet()**

```
bool gslc_PageFlipGet (
    gslc_tsGui * pGui )
```

Get state of pending page flip state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

True if screen requires page flip

**7.17.2.7 gslc\_PageFlipGo()**

```
void gslc_PageFlipGo (
    gslc_tsGui * pGui )
```

Update the visible screen if page has been marked for flipping.

- On some hardware this can trigger a double-buffering page flip.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

None

**7.17.2.8 gslc\_PageFlipSet()**

```
void gslc_PageFlipSet (
    gslc_tsGui * pGui,
    bool bNeeded )
```

Indicate whether the screen requires page flip.

- This is generally called with `bNeeded=true` whenever drawing has been done to the active page. Page flip is actually performed later when calling `PageFlipGo()`.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>bNeeded</i>	True if screen requires page flip

**Returns**

None

**7.17.2.9 gslc\_PageRedrawCalc()**

```
void gslc_PageRedrawCalc (
    gslc_tsGui * pGui )
```

Perform a redraw calculation on the page to determine if additional elements should also be redrawn.

This routine checks to see if any transparent elements have been marked as needing redraw. If so, the whole page may be marked as needing redraw (or at least the other elements that have been exposed underneath).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**7.17.2.10 gslc\_PageRedrawGo()**

```
void gslc_PageRedrawGo (
    gslc_tsGui * pGui )
```

Redraw all elements on the active page.

Only the elements that have been marked as needing redraw are rendered unless the entire page has been marked as needing redraw (in which case everything is drawn)

**Parameters**

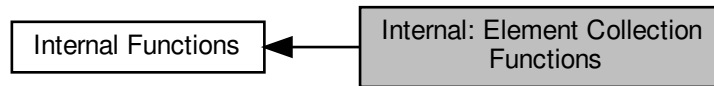
in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

## 7.18 Internal: Element Collection Functions

Collaboration diagram for Internal: Element Collection Functions:



### Functions

- void `gslc_CollectReset` (`gslc_tsCollect` \*pCollect, `gslc_tsElem` \*asElem, uint16\_t nElemMax, `gslc_tsElemRef` \*asElemRef, uint16\_t nElemRefMax)  
*Reset the members of an element collection.*
- `gslc_tsElemRef` \* `gslc_CollectElemAdd` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, const `gslc_tsElem` \*pElem, `gslc_teElemRefFlags` eFlags)  
*Add an element to a collection.*
- bool `gslc_CollectGetRedraw` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Determine if any elements in a collection need redraw.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemById` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, int16\_t nElemId)  
*Find an element in a collection by its Element ID.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemFromCoord` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, int16\_t nX, int16\_t nY, int16\_t \*pnElemInd)  
*Find an element in a collection by a coordinate coordinate.*
- int `gslc_CollectGetNextId` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Allocate the next available Element ID in a collection.*
- `gslc_tsElemRef` \* `gslc_CollectGetElemRefTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Get the element within a collection that is currently being tracked.*
- void `gslc_CollectSetElemTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRef, int16\_t nElemInd)  
*Set the element within a collection that is currently being tracked.*
- void `gslc_CollectSetParent` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRefParent)  
*Assign the parent element reference to all elements within a collection.*

### 7.18.1 Detailed Description

### 7.18.2 Function Documentation

7.18.2.1 `gslc_CollectElemAdd()`

```
gslc_tsElemRef* gslc_CollectElemAdd (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    const gslc_tsElem * pElem,
    gslc_teElemRefFlags eFlags )
```

Add an element to a collection.

- Note that the contents of `pElem` are copied to the collection's element array so the `pElem` pointer can be discarded after the call is complete.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>pElem</i>	Ptr to the element to add
in	<i>eFlags</i>	Flags describing the element (eg. whether the element should be stored in internal RAM array or is located in Flash/PROGMEM).

## Returns

Pointer to the element reference in the collection that has been added or NULL if there was an error

7.18.2.2 `gslc_CollectFindElemById()`

```
gslc_tsElemRef* gslc_CollectFindElemById (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    int16_t nElemId )
```

Find an element in a collection by its Element ID.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>nElemId</i>	Element ID to search for

## Returns

Pointer to the element reference in the collection that was found or NULL if no matches found

### 7.18.2.3 gslc\_CollectFindElemFromCoord()

```
gslc_tsElemRef* gslc_CollectFindElemFromCoord (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    int16_t nX,
    int16_t nY,
    int16_t * pnElemInd )
```

Find an element in a collection by a coordinate coordinate.

- A match is found if the element is "clickable" (bClickEn=true) and the coordinate falls within the element's bounds (rElem).

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>nX</i>	Absolute X coordinate to use for search
in	<i>nY</i>	Absolute Y coordinate to use for search
out	<i>pnElemInd</i>	Pointer to element index found

#### Returns

Pointer to the element reference in the collection that was found or NULL if no matches found

### 7.18.2.4 gslc\_CollectGetElemRefTracked()

```
gslc_tsElemRef* gslc_CollectGetElemRefTracked (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect )
```

Get the element within a collection that is currently being tracked.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection

#### Returns

Pointer to the element reference in the collection that is currently being tracked or NULL if no elements are being tracked



### 7.18.2.5 `gslc_CollectGetNextId()`

```
int gslc_CollectGetNextId (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect )
```

Allocate the next available Element ID in a collection.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection

#### Returns

Element ID that is reserved for use

### 7.18.2.6 `gslc_CollectGetRedraw()`

```
bool gslc_CollectGetRedraw (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect )
```

Determine if any elements in a collection need redraw.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to Element collection

#### Returns

True if redraw required, false otherwise

### 7.18.2.7 `gslc_CollectReset()`

```
void gslc_CollectReset (
    gslc_tsCollect * pCollect,
    gslc_tsElem * asElem,
    uint16_t nElemMax,
    gslc_tsElemRef * asElemRef,
    uint16_t nElemRefMax )
```

Reset the members of an element collection.

**Parameters**

in	<i>pCollect</i>	Pointer to the collection
in	<i>asElem</i>	Internal element array storage to associate with the collection
in	<i>nElemMax</i>	Maximum number of elements that can be added to the internal element array (ie. RAM))
in	<i>asElemRef</i>	Internal element reference array storage to associate with the collection. All elements, whether they are located in the internal element array or in external Flash (PROGMEM) storage, require an entry in the element reference array.
in	<i>nElemRefMax</i>	Maximum number of elements in the reference array. This is effectively the maximum number of elements that can appear in the collection, irrespective of whether it is stored in RAM or Flash (PROGMEM).

**Returns**

none

**7.18.2.8 gslc\_CollectSetElemTracked()**

```
void gslc_CollectSetElemTracked (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    gslc_tsElemRef * pElemRef,
    int16_t nElemInd )
```

Set the element within a collection that is currently being tracked.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>pElemRef</i>	Ptr to element reference to mark as being tracked
in	<i>nElemInd</i>	Element index to mark as being tracked

**Returns**

none

**7.18.2.9 gslc\_CollectSetParent()**

```
void gslc_CollectSetParent (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    gslc_tsElemRef * pElemRefParent )
```

Assign the parent element reference to all elements within a collection.

- This is generally used in the case of compound elements where updates to a sub-element should cause the parent (compound element) to be redrawn as well.)

**Parameters**

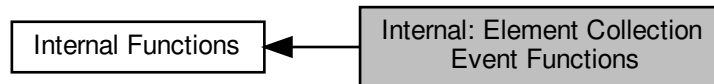
in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>pElemRefParent</i>	Ptr to element reference that is the parent

**Returns**

none

## 7.19 Internal: Element Collection Event Functions

Collaboration diagram for Internal: Element Collection Event Functions:



### Functions

- `bool gslc_CollectEvent (void *pvGui, gslc\_tsEvent sEvent)`  
Common event handler function for an element collection.
- `void gslc\_CollectTouch (gslc\_tsGui *pGui, gslc\_tsCollect *pCollect, gslc\_tsEventTouch *pEventTouch)`  
Handle touch events within the element collection.
- `bool gslc\_CollectTouchCompound (void *pvGui, void *pvElemRef, gslc\_teTouch eTouch, int16_t nRelX, int16_t nRelY, gslc\_tsCollect *pCollect)`  
Handle dispatch of touch (up,down,move) events to compound elements sub elements.
- `void gslc\_CollectInput (gslc\_tsGui *pGui, gslc\_tsCollect *pCollect, gslc\_tsEventTouch *pEventTouch)`  
Handle direct input events within the element collection.

### 7.19.1 Detailed Description

### 7.19.2 Function Documentation

#### 7.19.2.1 `gslc_CollectEvent()`

```
bool gslc_CollectEvent (
    void * pvGui,
    gslc\_tsEvent sEvent )
```

Common event handler function for an element collection.

#### Parameters

in	<i>pvGui</i>	Void pointer to GUI
in	<i>sEvent</i>	Event data structure

**Returns**

true if success, false if fail

**7.19.2.2 gslc\_CollectInput()**

```
void gslc_CollectInput (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    gslc_tsEventTouch * pEventTouch )
```

Handle direct input events within the element collection.

**Parameters**

in	<i>pGui</i>	Pointer to the GUI
in	<i>pCollect</i>	Ptr to the element collection
in	<i>pEventTouch</i>	Ptr to the touch event structure

**Returns**

none

**7.19.2.3 gslc\_CollectTouch()**

```
void gslc_CollectTouch (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    gslc_tsEventTouch * pEventTouch )
```

Handle touch events within the element collection.

**Parameters**

in	<i>pGui</i>	Pointer to the GUI
in	<i>pCollect</i>	Ptr to the element collection
in	<i>pEventTouch</i>	Ptr to the touch event structure

**Returns**

none

#### 7.19.2.4 gslc\_CollectTouchCompound()

```
bool gslc_CollectTouchCompound (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY,
    gslc_tsCollect * pCollect )
```

Handle dispatch of touch (up,down,move) events to compound elements sub elements.

##### Parameters

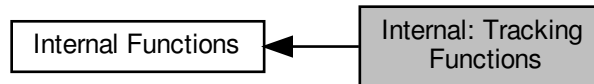
in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element Reference(typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element
in	<i>pCollect</i>	Collection containing sub elements

##### Returns

true if success, false otherwise

## 7.20 Internal: Tracking Functions

Collaboration diagram for Internal: Tracking Functions:



### Functions

- void `gslc_TrackTouch` (`gslc_tsGui` \*pGui, `gslc_tsPage` \*pPage, int16\_t nX, int16\_t nY, uint16\_t nPress)  
*Handles a touch event and performs the necessary tracking, glowing and selection actions depending on the press state.*
- void `gslc_TrackInput` (`gslc_tsGui` \*pGui, `gslc_teInputRawEvent` eInputEvent, int16\_t nInputVal)  
*Handles a direct input event and performs the necessary tracking, glowing and selection actions depending on the state.*
- bool `gslc_InputMapLookup` (`gslc_tsGui` \*pGui, `gslc_teInputRawEvent` eInputEvent, int16\_t nInputVal, `gslc_teAction` \*peAction, int16\_t \*pnActionVal)

### 7.20.1 Detailed Description

### 7.20.2 Function Documentation

#### 7.20.2.1 `gslc_InputMapLookup()`

```

bool gslc_InputMapLookup (
    gslc_tsGui * pGui,
    gslc_teInputRawEvent eInputEvent,
    int16_t nInputVal,
    gslc_teAction * peAction,
    int16_t * pnActionVal )
  
```

**Todo** Doc.

This API is experimental and subject to change

Convert an external input event into a GUI action

- Use the InputMap table to determine the action to take as a result of the external input event.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>eInputEvent</i>	Indication of event type
in	<i>nInputVal</i>	Additional data for event type
out	<i>peAction</i>	The GUI action to take
out	<i>pnActionVal</i>	Additional parameter for the GUI action

**Returns**

true if a matching event was found, false if none

**7.20.2.2 gslc\_TrackInput()**

```
void gslc_TrackInput (
    gslc_tsGui * pGui,
    gslc_teInputRawEvent eInputEvent,
    int16_t nInputVal )
```

Handles a direct input event and performs the necessary tracking, glowing and selection actions depending on the state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>eInputEvent</i>	Indication of event type
in	<i>nInputVal</i>	Additional data for event type

**Returns**

none

**7.20.2.3 gslc\_TrackTouch()**

```
void gslc_TrackTouch (
    gslc_tsGui * pGui,
    gslc_tsPage * pPage,
    int16_t nX,
    int16_t nY,
    uint16_t nPress )
```

Handles a touch event and performs the necessary tracking, glowing and selection actions depending on the press state.



**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pPage</i>	Pointer to current page
in	<i>nX</i>	X coordinate of touch event
in	<i>nY</i>	Y coordinate of touch event
in	<i>nPress</i>	Pressure level of touch event (0 for none, else touch)

**Returns**

none

## 7.21 Internal: Cleanup Functions

Collaboration diagram for Internal: Cleanup Functions:



### Functions

- void `gslc_GuiDestruct` (`gslc_tsGui` \*pGui)  
*Free up any surfaces associated with the GUI, pages, collections and elements.*
- void `gslc_PageDestruct` (`gslc_tsGui` \*pGui, `gslc_tsPage` \*pPage)  
*Free up any members associated with a page.*
- void `gslc_CollectDestruct` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Free up any members associated with an element collection.*
- void `gslc_ElemDestruct` (`gslc_tsElem` \*pElem)  
*Free up any members associated with an element.*
- void `gslc_ResetFont` (`gslc_tsFont` \*pFont)  
*Initialize a Font struct.*
- void `gslc_ResetElem` (`gslc_tsElem` \*pElem)  
*Initialize an Element struct.*

### 7.21.1 Detailed Description

### 7.21.2 Function Documentation

#### 7.21.2.1 `gslc_CollectDestruct()`

```
void gslc_CollectDestruct (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect )
```

Free up any members associated with an element collection.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to collection

**Returns**

none

**7.21.2.2 `gslc_ElemDestruct()`**

```
void gslc_ElemDestruct (
    gslc_tsElem * pElem )
```

Free up any members associated with an element.

**Parameters**

in	<i>pElem</i>	Pointer to element
----	--------------	--------------------

**Returns**

none

**7.21.2.3 `gslc_GuiDestruct()`**

```
void gslc_GuiDestruct (
    gslc_tsGui * pGui )
```

Free up any surfaces associated with the GUI, pages, collections and elements.

Also frees up any fonts.

- Called by `gslc_Quit()`

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**7.21.2.4 `gslc_PageDestruct()`**

```
void gslc_PageDestruct (
    gslc_tsGui * pGui,
    gslc_tsPage * pPage )
```

Free up any members associated with a page.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pPage</i>	Pointer to Page

## Returns

none

## 7.21.2.5 gslc\_ResetElem()

```
void gslc_ResetElem (
    gslc_tsElem * pElem )
```

Initialize an Element struct.

## Parameters

in	<i>pElem</i>	Pointer to Element
----	--------------	--------------------

## Returns

none

## 7.21.2.6 gslc\_ResetFont()

```
void gslc_ResetFont (
    gslc_tsFont * pFont )
```

Initialize a Font struct.

## Parameters

in	<i>pFont</i>	Pointer to Font
----	--------------	-----------------

## Returns

none



## Chapter 8

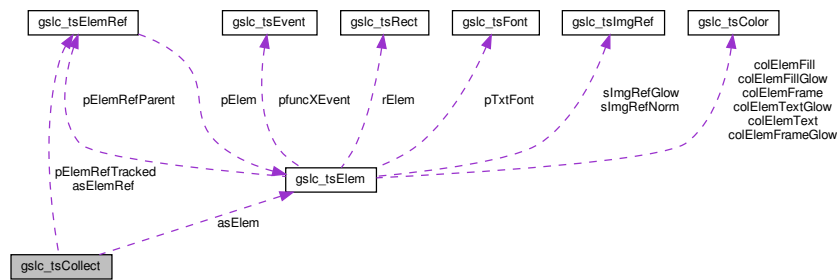
# Data Structure Documentation

### 8.1 gslc\_tsCollect Struct Reference

Element collection struct.

```
#include <GUIslice.h>
```

Collaboration diagram for gslc\_tsCollect:



#### Data Fields

- [gslc\\_tsElem](#) \* [asElem](#)  
*Array of elements.*
- [uint16\\_t](#) [nElemMax](#)  
*Maximum number of elements to allocate (in RAM)*
- [uint16\\_t](#) [nElemCnt](#)  
*Number of elements allocated.*
- [int16\\_t](#) [nElemAutoldNext](#)  
*Next Element ID for auto-assignment.*
- [gslc\\_tsElemRef](#) \* [asElemRef](#)  
*Array of element references.*
- [uint16\\_t](#) [nElemRefMax](#)  
*Maximum number of element references to allocate.*
- [uint16\\_t](#) [nElemRefCnt](#)  
*Number of element references allocated.*
- [gslc\\_tsElemRef](#) \* [pElemRefTracked](#)  
*Element reference currently being touch-tracked (NULL for none)*
- [int16\\_t](#) [nElemIndTracked](#)  
*Element index currently being touch-tracked (GSLC\_IND\_NONE for none)*

### 8.1.1 Detailed Description

Element collection struct.

- Collections are used to maintain a list of elements and any touch tracking status.
- Pages and Compound Elements both instantiate a Collection

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

- [src/GUISlice.h](#)

## 8.2 gslc\_tsColor Struct Reference

Color structure. Defines RGB triplet.

```
#include <GUISlice.h>
```

### Data Fields

- `uint8_t r`  
*RGB red value.*
- `uint8_t g`  
*RGB green value.*
- `uint8_t b`  
*RGB blue value.*

### 8.2.1 Detailed Description

Color structure. Defines RGB triplet.

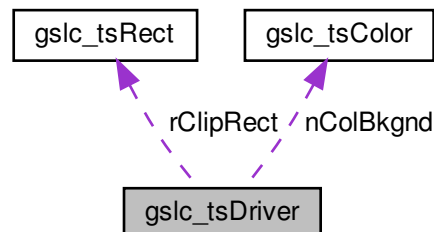
The documentation for this struct was generated from the following file:

- [src/GUISlice.h](#)

## 8.3 gslc\_tsDriver Struct Reference

```
#include <GUISlice_drv_adagfx.h>
```

Collaboration diagram for `gslc_tsDriver`:





## Data Fields

- [gslc\\_tsColor nColBkgnd](#)  
*Background color (if not image-based)*
- [gslc\\_tsRect rClipRect](#)  
*Clipping rectangle.*
- `const void *` [pvFontLast](#)  
*Last loadFont() reference.*

### 8.3.1 Field Documentation

#### 8.3.1.1 nColBkgnd

[gslc\\_tsColor](#) `gslc_tsDriver::nColBkgnd`

Background color (if not image-based)

#### 8.3.1.2 pvFontLast

`const void*` `gslc_tsDriver::pvFontLast`

Last loadFont() reference.

#### 8.3.1.3 rClipRect

[gslc\\_tsRect](#) `gslc_tsDriver::rClipRect`

Clipping rectangle.

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

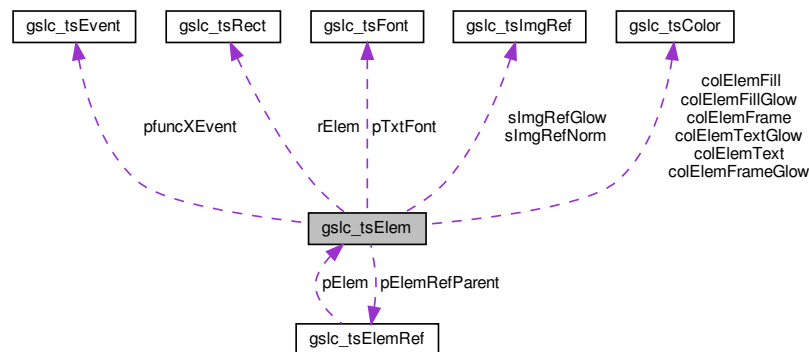
- [src/GUISlice\\_drv\\_adagfx.h](#)
- [src/GUISlice\\_drv\\_m5stack.h](#)
- [src/GUISlice\\_drv\\_sdl.h](#)
- [src/GUISlice\\_drv\\_tft\\_espi.h](#)
- [src/GUISlice\\_drv\\_utft.h](#)

## 8.4 gslc\_tsElem Struct Reference

Element Struct.

```
#include <GUIslice.h>
```

Collaboration diagram for gslc\_tsElem:



### Data Fields

- `int16_t nId`  
*Element ID specified by user.*
- `uint16_t nFeatures`  
*Element feature vector (appearance/behavior)*
- `int16_t nType`  
*Element type enumeration.*
- `gslc_tsRect rElem`  
*Rect region containing element.*
- `int16_t nGroup`  
*Group ID that the element belongs to.*
- `gslc_tsColor colElemFrame`  
*Color for frame.*
- `gslc_tsColor colElemFill`  
*Color for background fill.*
- `gslc_tsColor colElemFrameGlow`  
*Color to use for frame when glowing.*
- `gslc_tsColor colElemFillGlow`  
*Color to use for fill when glowing.*
- `gslc_tsImgRef sImgRefNorm`  
*Image reference to draw (normal)*
- `gslc_tsImgRef sImgRefGlow`  
*Image reference to draw (glowing)*
- `gslc_tsElemRef * pElemRefParent`  
*Parent element reference.*
- `char * pStrBuf`

- Ptr to text string buffer to overlay.*
- [uint8\\_t nStrBufMax](#)  
*Size of string buffer.*
- [gslc\\_teTxtFlags eTxtFlags](#)  
*Flags associated with text buffer.*
- [gslc\\_tsColor colElemText](#)  
*Color of overlay text.*
- [gslc\\_tsColor colElemTextGlow](#)  
*Color of overlay text when glowing.*
- [int8\\_t eTxtAlign](#)  
*Alignment of overlay text.*
- [int8\\_t nTxtMarginX](#)  
*Margin of overlay text within rect region (x offset)*
- [int8\\_t nTxtMarginY](#)  
*Margin of overlay text within rect region (y offset)*
- [gslc\\_tsFont \\* pTxtFont](#)  
*Ptr to Font for overlay text.*
- [void \\* pXData](#)  
*Ptr to extended data structure.*
- [GSLC\\_CB\\_EVENT pfuncXEvent](#)  
*UNUSED: Callback func ptr for event tree (draw,touch,tick)*
- [GSLC\\_CB\\_DRAW pfuncXDraw](#)  
*Callback func ptr for custom drawing.*
- [GSLC\\_CB\\_TOUCH pfuncXTouch](#)  
*Callback func ptr for touch.*
- [GSLC\\_CB\\_TICK pfuncXTick](#)  
*Callback func ptr for timer/main loop tick.*

### 8.4.1 Detailed Description

Element Struct.

- Represents a single graphic element in the GUIslice environment
- A page is made up of a number of elements
- Each element is created with a user-specified ID for further accesses (or `GSLC_ID_AUTO` for it to be auto-generated)
- Display order of elements in a page is based upon the creation order
- Extensions to the core element types is provided through the `pXData` reference and `pfuncX*` callback functions.

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

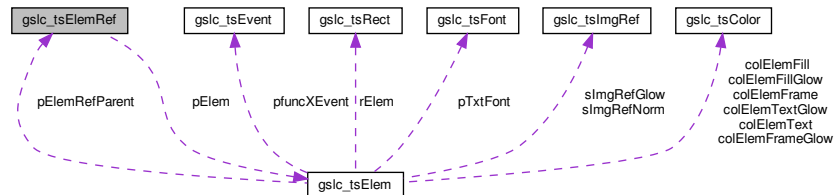
- [src/GUIslice.h](#)

## 8.5 gslc\_tsElemRef Struct Reference

Element reference structure.

```
#include <GUIslice.h>
```

Collaboration diagram for gslc\_tsElemRef:



### Data Fields

- [gslc\\_tsElem](#) \* [pElem](#)  
*Pointer to element in memory [RAM,FLASH].*
- [gslc\\_teElemRefFlags](#) [eElemFlags](#)  
*Element reference flags.*

### 8.5.1 Detailed Description

Element reference structure.

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

- [src/GUIslice.h](#)

## 8.6 gslc\_tsEvent Struct Reference

Event structure.

```
#include <GUIslice.h>
```

### Data Fields

- [gslc\\_teEventType](#) [eType](#)  
*Event type.*
- [uint8\\_t](#) [nSubType](#)  
*Event sub-type.*
- [void](#) \* [pvScope](#)  
*Event target scope (eg. Page,Collection,Event)*
- [void](#) \* [pvData](#)  
*Generic data pointer for event.*

### 8.6.1 Detailed Description

Event structure.

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

- [src/GUISlice.h](#)

## 8.7 gslc\_tsEventTouch Struct Reference

Structure used to pass touch data through event.

```
#include <GUISlice.h>
```

### Data Fields

- [gslc\\_teTouch eTouch](#)  
*Touch state.*
- [int16\\_t nX](#)  
*Touch X coordinate (or param1)*
- [int16\\_t nY](#)  
*Touch Y coordinate (or param2)*

### 8.7.1 Detailed Description

Structure used to pass touch data through event.

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

- [src/GUISlice.h](#)

## 8.8 gslc\_tsFont Struct Reference

Font reference structure.

```
#include <GUISlice.h>
```

### Data Fields

- [int16\\_t nId](#)  
*Font ID specified by user.*
- [gslc\\_teFontRefType eFontRefType](#)  
*Font reference type.*
- [gslc\\_teFontRefMode eFontRefMode](#)  
*Font reference mode.*
- `const void *` [pvFont](#)  
*Void ptr to the font reference (type defined by driver)*
- [uint16\\_t nSize](#)  
*Font size.*



- *Adafruit GFX Touch Flip x axis.*  
• uint8\_t [nFlipY](#)
- *Adafruit GFX Touch Flip x axis.*  
• int16\_t [nTouchCalXMin](#)
- *Calibration X minimum reading.*  
• int16\_t [nTouchCalXMax](#)
- *Calibration X maximum reading.*  
• int16\_t [nTouchCalYMin](#)
- *Calibration Y minimum reading.*  
• int16\_t [nTouchCalYMax](#)
- *Calibration Y maximum reading.*  
• int16\_t [nTouchCalPressMin](#)
- *Calibration minimum pressure threshold.*  
• int16\_t [nTouchCalPressMax](#)
- *Calibration maximum pressure threshold.*  
• [gslc\\_tsFont](#) \* [asFont](#)
- *Collection of loaded fonts.*  
• uint8\_t [nFontMax](#)
- *Maximum number of fonts to allocate.*  
• uint8\_t [nFontCnt](#)
- *Number of fonts allocated.*  
• uint8\_t [nRoundRadius](#)
- *Radius for rounded elements.*  
• [gslc\\_tsColor](#) [sTransCol](#)
- *Color used for transparent image regions (GSLC\_BMP\_TRANS\_EN=1)*  
• [gslc\\_tsElem](#) [sElemTmp](#)
- *Temporary element.*  
• [gslc\\_tsElemRef](#) [sElemRefTmp](#)
- *Temporary element reference.*  
• [gslc\\_tsElem](#) [sElemTmpProg](#)
- *Temporary element for Flash compatibility.*  
• [gslc\\_telnitStat](#) [elnitStatTouch](#)
- *Status of touch initialization.*  
• int16\_t [nTouchLastX](#)
- *Last touch event X coord.*  
• int16\_t [nTouchLastY](#)
- *Last touch event Y coord.*  
• uint16\_t [nTouchLastPress](#)
- *Last touch event pressure (0=none)*  
• bool [bTouchRemapEn](#)
- *Enable touch remapping?*  
• bool [bTouchRemapYX](#)
- *Enable touch controller swapping of X & Y.*  
• bool [bTouchEn](#)
- *Enable reaction to touch events.*  
• void \* [pvDriver](#)
- *Driver-specific members (gslc\_tsDriver\*)*  
• bool [bRedrawNeeded](#)
- *Does anything on page require redraw?*  
• bool [bRedrawPartialEn](#)
- *Driver supports partial page redraw.*

- [bool bEventPending](#)  
*Is there an event pending?*
- [gslc\\_tsEventTouch sEventTouchPend](#)  
*A touch event that has been deferred (if bEventPending=true)*
- [gslc\\_tsEvent sEventPend](#)  
*An event that has been deferred (if bEventPending=true)*
- [gslc\\_tsImgRef sImgRefBkgnd](#)  
*Image reference for background.*
- [uint8\\_t nFrameRateCnt](#)  
*Diagnostic frame rate count.*
- [uint8\\_t nFrameRateStart](#)  
*Diagnostic frame rate timestamp.*
- [gslc\\_tsPage \\* asPage](#)  
*Array of all pages defined in system.*
- [uint8\\_t nPageMax](#)  
*Maximum number of pages that can be defined.*
- [uint8\\_t nPageCnt](#)  
*Current number of pages defined.*
- [gslc\\_tsPage \\* apPageStack \[GSLC\\_STACK\\_\\_MAX\]](#)  
*Stack of pages.*
- [bool abPageStackActive \[GSLC\\_STACK\\_\\_MAX\]](#)  
*Whether page in stack can receive touch events.*
- [bool abPageStackDoDraw \[GSLC\\_STACK\\_\\_MAX\]](#)  
*Whether page in stack is still actively drawn.*
- [bool bScreenNeedRedraw](#)  
*Screen requires a redraw.*
- [bool bScreenNeedFlip](#)  
*Screen requires a page flip.*
- [bool bInvalidateEn](#)  
*A region of the display has been invalidated.*
- [gslc\\_tsRect rInvalidateRect](#)  
*The rect region that has been invalidated.*
- [GSLC\\_CB\\_PIN\\_POLL pfuncPinPoll](#)  
*Callback func ptr for pin polling.*
- [gslc\\_tsInputMap \\* asInputMap](#)  
*Array of input maps.*
- [uint8\\_t nInputMapMax](#)  
*Maximum number of input maps.*
- [uint8\\_t nInputMapCnt](#)  
*Current number of input maps.*
- [uint8\\_t nInputMode](#)  
*Input mode: 0=navigate, 1=edit.*
- [int16\\_t nFocusPageInd](#)  
*Index of page in stack currently in focus.*
- [gslc\\_tsPage \\* pFocusPage](#)  
*Page ptr currently in focus.*
- [gslc\\_tsElemRef \\* pFocusElemRef](#)  
*Reference to element in focus.*
- [int16\\_t nFocusElemInd](#)  
*Index of element in page currently in focus.*
- [int16\\_t nFocusElemMax](#)



- Max number of elements in page in focus.*
- [gslc\\_tsColor colFocusNone](#)
  - Focus frame color when not in focus (typically background color)*
- [gslc\\_tsColor colFocus](#)
  - Focus frame color when in focus.*
- [gslc\\_tsColor colFocusEdit](#)
  - Focus frame color when in focus and edit mode.*
- `int16_t nFocusSavedPageInd`
  - Focus page index saved prior to popup/overlay.*
- `int16_t nFocusSavedElemInd`
  - Focus element index saved prior to popup/overlay.*

### 8.9.1 Detailed Description

GUI structure.

- Contains all GUI state and content
- Maintains list of one or more pages

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

- [src/GUISlice.h](#)

## 8.10 gslc\_tsImgRef Struct Reference

Image reference structure.

```
#include <GUISlice.h>
```

### Data Fields

- `const unsigned char * pImgBuf`
  - Pointer to input image buffer in memory [RAM,FLASH].*
- `const char * pFname`
  - Pathname to input image file [FILE,SD].*
- [gslc\\_telmgRefFlags elmgFlags](#)
  - Image reference flags.*
- `void * pvImgRaw`
  - Ptr to raw output image data (for pre-loaded images)*

### 8.10.1 Detailed Description

Image reference structure.

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

- [src/GUISlice.h](#)

## 8.11 gslc\_tsInputMap Struct Reference

Input mapping.

```
#include <GUIslice.h>
```

### Data Fields

- [gslc\\_telInputRawEvent eEvent](#)  
*The input event.*
- [int16\\_t nVal](#)  
*The value associated with the input event.*
- [gslc\\_teAction eAction](#)  
*Resulting action.*
- [int16\\_t nActionVal](#)  
*The value for the output action.*

### 8.11.1 Detailed Description

Input mapping.

- Describes mapping from keyboard or GPIO input to a GUI action (such as changing the current element focus)
- This is generally used to support keyboard or GPIO control over the GUI operation

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

- [src/GUIslice.h](#)

## 8.12 gslc\_tsKey Struct Reference

Key information. Defines everything we need to know about a particular key.

```
#include <XKeyPad.h>
```

### Data Fields

- [uint8\\_t nId](#)  
*Unique identifier.*
- [uint8\\_t nRow](#)  
*Row to place the key (0 is top-most)*
- [uint8\\_t nCol](#)  
*Column to place the key (0 is left-most)*
- [uint8\\_t nRowSpan](#)  
*Number of rows that key takes up (in units of nButtonSzH pixels)*
- [uint8\\_t nColSpan](#)  
*Number of columns that key takes up (in units of nButtonSzW pixels)*
- [uint8\\_t nType](#)  
*Key type.*

### 8.12.1 Detailed Description

Key information. Defines everything we need to know about a particular key.

### 8.12.2 Field Documentation

#### 8.12.2.1 nCol

```
uint8_t gslc_tsKey::nCol
```

Column to place the key (0 is left-most)

#### 8.12.2.2 nColSpan

```
uint8_t gslc_tsKey::nColSpan
```

Number of columns that key takes up (in units of nButtonSzW pixels)

#### 8.12.2.3 nId

```
uint8_t gslc_tsKey::nId
```

Unique identifier.

#### 8.12.2.4 nRow

```
uint8_t gslc_tsKey::nRow
```

Row to place the key (0 is top-most)

#### 8.12.2.5 nRowSpan

```
uint8_t gslc_tsKey::nRowSpan
```

Number of rows that key takes up (in units of nButtonSzH pixels)

#### 8.12.2.6 nType

```
uint8_t gslc_tsKey::nType
```

Key type.

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

- [src/elem/XKeyPad.h](#)

### 8.13 gslc\_tsLabelSpecial Struct Reference

Key Label strings for special buttons.

```
#include <XKeyPad.h>
```

#### Data Fields

- [uint8\\_t nId](#)
- [const char \\* pLabel](#)  
*Unique key ID.*

#### 8.13.1 Detailed Description

Key Label strings for special buttons.

- Includes ID and string

#### 8.13.2 Field Documentation

##### 8.13.2.1 nId

```
uint8_t gslc_tsLabelSpecial::nId
```

##### 8.13.2.2 pLabel

```
const char* gslc_tsLabelSpecial::pLabel
```

Unique key ID.

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

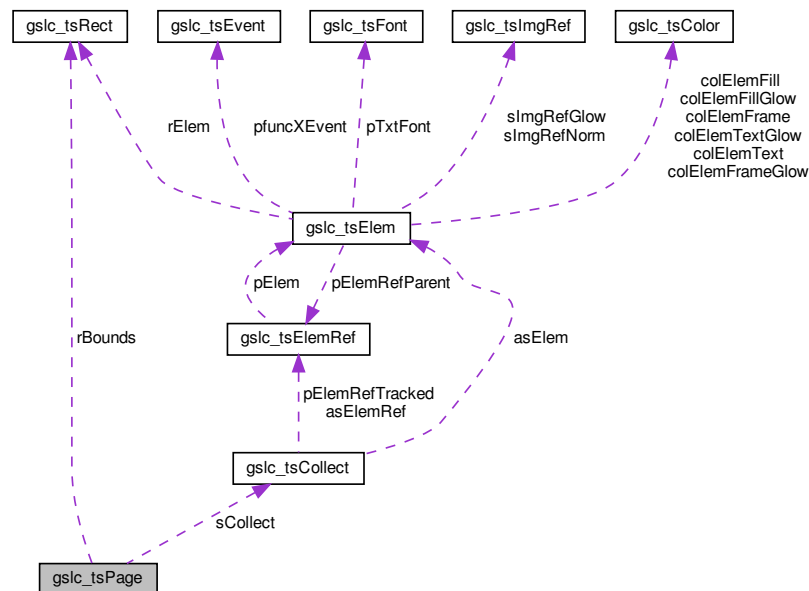
- [src/elem/XKeyPad.h](#)

## 8.14 gslc\_tsPage Struct Reference

Page structure.

```
#include <GUIslice.h>
```

Collaboration diagram for gslc\_tsPage:



### Data Fields

- [gslc\\_tsCollect sCollect](#)  
*Collection of elements on page.*
- `int16_t nPageId`  
*Page identifier.*
- [gslc\\_tsRect rBounds](#)  
*Bounding rect for page elements.*

#### 8.14.1 Detailed Description

Page structure.

- A page contains a collection of elements
- Many redraw functions operate at a page level
- Maintains state as to whether redraw or screen flip is required

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

- [src/GUIslice.h](#)

## 8.15 gslc\_tsPt Struct Reference

Define point coordinates.

```
#include <GUIslice.h>
```

### Data Fields

- `int16_t x`  
*X coordinate.*
- `int16_t y`  
*Y coordinate.*

### 8.15.1 Detailed Description

Define point coordinates.

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

- `src/GUIslice.h`

## 8.16 gslc\_tsRect Struct Reference

Rectangular region. Defines X,Y corner coordinates plus dimensions.

```
#include <GUIslice.h>
```

### Data Fields

- `int16_t x`  
*X coordinate of corner.*
- `int16_t y`  
*Y coordinate of corner.*
- `uint16_t w`  
*Width of region.*
- `uint16_t h`  
*Height of region.*

### 8.16.1 Detailed Description

Rectangular region. Defines X,Y corner coordinates plus dimensions.

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

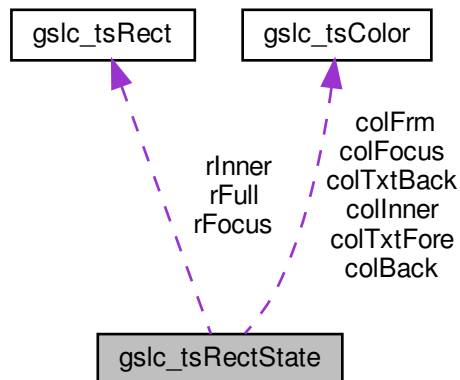
- `src/GUIslice.h`

## 8.17 gslc\_tsRectState Struct Reference

State associated with an element's region.

```
#include <GUIslice.h>
```

Collaboration diagram for gslc\_tsRectState:



### Data Fields

- [gslc\\_tsRect rFocus](#)
- [gslc\\_tsRect rFull](#)
- [gslc\\_tsRect rInner](#)
- [gslc\\_tsColor colFocus](#)
- [gslc\\_tsColor colFrm](#)
- [gslc\\_tsColor colInner](#)
- [gslc\\_tsColor colBack](#)
- [gslc\\_tsColor colTxtFore](#)
- [gslc\\_tsColor colTxtBack](#)

### 8.17.1 Detailed Description

State associated with an element's region.

- This struct is used for [gslc\\_ElemCalcRectState\(\)](#)
- Accounts for various rects including focus, frame and internal content
- Also contains the various colors associated with each region.

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

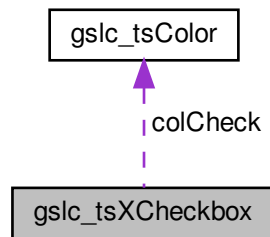
- [src/GUIslice.h](#)

## 8.18 gslc\_tsXCheckbox Struct Reference

Extended data for Checkbox element.

```
#include <XCheckbox.h>
```

Collaboration diagram for gslc\_tsXCheckbox:



### Data Fields

- bool `bRadio`  
*Radio-button operation if true.*
- `gslc_teXCheckboxStyle` `nStyle`  
*Drawing style for element.*
- bool `bChecked`  
*Indicates if it is selected (checked)*
- `gslc_tsColor` `colCheck`  
*Color of checked inner fill.*
- `GSFC_CB_XCHECKBOX` `pfuncXToggle`  
*Callback event to say element has changed.*

### 8.18.1 Detailed Description

Extended data for Checkbox element.

### 8.18.2 Field Documentation

#### 8.18.2.1 bChecked

```
bool gslc_tsXCheckbox::bChecked
```

Indicates if it is selected (checked)



## 8.18.2.2 bRadio

```
bool gslc_tsXCheckbox::bRadio
```

Radio-button operation if true.

## 8.18.2.3 colCheck

```
gslc_tsColor gslc_tsXCheckbox::colCheck
```

Color of checked inner fill.

## 8.18.2.4 nStyle

```
gslc_tsXCheckboxStyle gslc_tsXCheckbox::nStyle
```

Drawing style for element.

## 8.18.2.5 pfuncXToggle

```
GSLC_CB_XCHECKBOX gslc_tsXCheckbox::pfuncXToggle
```

Callback event to say element has changed.

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

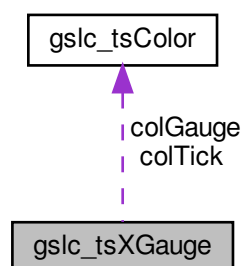
- [src/elm/XCheckbox.h](#)

## 8.19 gslc\_tsXGauge Struct Reference

Extended data for Gauge element.

```
#include <XGauge.h>
```

Collaboration diagram for gslc\_tsXGauge:



## Data Fields

- `int16_t nMin`  
*Minimum control value.*
- `int16_t nMax`  
*Maximum control value.*
- `int16_t nVal`  
*Current control value.*
- `int16_t nValLast`  
*Last value.*
- `bool bValLastValid`  
*Last value valid?*
- `gslc_tsXGaugeStyle nStyle`  
*Gauge sub-type.*
- `gslc_tsColor colGauge`  
*Color of gauge fill bar.*
- `gslc_tsColor colTick`  
*Color of gauge tick marks.*
- `uint16_t nTickCnt`  
*Number of gauge tick marks.*
- `uint16_t nTickLen`  
*Length of gauge tick marks.*
- `bool bVert`  
*Vertical if true, else Horizontal.*
- `bool bFlip`  
*Reverse direction of gauge.*
- `uint16_t nIndicLen`  
*Indicator length.*
- `uint16_t nIndicTip`  
*Size of tip at end of indicator.*
- `bool bIndicFill`  
*Fill the indicator if true.*

### 8.19.1 Detailed Description

Extended data for Gauge element.

### 8.19.2 Field Documentation

#### 8.19.2.1 bFlip

```
bool gslc_tsXGauge::bFlip
```

Reverse direction of gauge.

### 8.19.2.2 bIndicFill

```
bool gslc_tsXGauge::bIndicFill
```

Fill the indicator if true.

### 8.19.2.3 bValLastValid

```
bool gslc_tsXGauge::bValLastValid
```

Last value valid?

### 8.19.2.4 bVert

```
bool gslc_tsXGauge::bVert
```

Vertical if true, else Horizontal.

### 8.19.2.5 colGauge

```
gslc_tsColor gslc_tsXGauge::colGauge
```

Color of gauge fill bar.

### 8.19.2.6 colTick

```
gslc_tsColor gslc_tsXGauge::colTick
```

Color of gauge tick marks.

### 8.19.2.7 nIndicLen

```
uint16_t gslc_tsXGauge::nIndicLen
```

Indicator length.

#### 8.19.2.8 nIndicTip

```
uint16_t gslc_tsXGauge::nIndicTip
```

Size of tip at end of indicator.

#### 8.19.2.9 nMax

```
int16_t gslc_tsXGauge::nMax
```

Maximum control value.

#### 8.19.2.10 nMin

```
int16_t gslc_tsXGauge::nMin
```

Minimum control value.

#### 8.19.2.11 nStyle

```
gslc_tXGaugeStyle gslc_tsXGauge::nStyle
```

Gauge sub-type.

#### 8.19.2.12 nTickCnt

```
uint16_t gslc_tsXGauge::nTickCnt
```

Number of gauge tick marks.

#### 8.19.2.13 nTickLen

```
uint16_t gslc_tsXGauge::nTickLen
```

Length of gauge tick marks.

## 8.19.2.14 nVal

```
int16_t gslc_tsXGauge::nVal
```

Current control value.

## 8.19.2.15 nValLast

```
int16_t gslc_tsXGauge::nValLast
```

Last value.

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

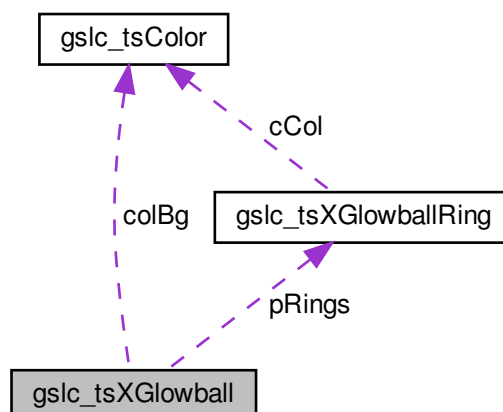
- [src/elm/XGauge.h](#)

## 8.20 gslc\_tsXGlowball Struct Reference

Extended data for Slider element.

```
#include <XGlowball.h>
```

Collaboration diagram for gslc\_tsXGlowball:



## Data Fields

- `int16_t nMidX`  
*Gauge midpoint X coord.*
- `int16_t nMidY`  
*Gauge midpoint Y coord.*
- `gslc_tsXGlowballRing * pRings`  
*Ring definition array.*
- `uint8_t nNumRings`  
*Number of rings in definition.*
- `uint16_t nQuality`  
*Rendering quality (number of segments / rotation)*
- `int16_t nAngStart`  
*Starting angle (0..510 degrees)*
- `int16_t nAngEnd`  
*Ending angle (0..510 degrees)*
- `gslc_tsColor colBg`  
*Background color (for redraw)*
- `int16_t nVal`  
*Current value.*
- `int16_t nValLast`  
*Previous value.*

### 8.20.1 Detailed Description

Extended data for Slider element.

### 8.20.2 Field Documentation

#### 8.20.2.1 colBg

```
gslc_tsColor gslc_tsXGlowball::colBg
```

Background color (for redraw)

#### 8.20.2.2 nAngEnd

```
int16_t gslc_tsXGlowball::nAngEnd
```

Ending angle (0..510 degrees)

### 8.20.2.3 nAngStart

```
int16_t gslc_tsXGlowball::nAngStart
```

Starting angle (0..510 degrees)

### 8.20.2.4 nMidX

```
int16_t gslc_tsXGlowball::nMidX
```

Gauge midpoint X coord.

### 8.20.2.5 nMidY

```
int16_t gslc_tsXGlowball::nMidY
```

Gauge midpoint Y coord.

### 8.20.2.6 nNumRings

```
uint8_t gslc_tsXGlowball::nNumRings
```

Number of rings in definition.

### 8.20.2.7 nQuality

```
uint16_t gslc_tsXGlowball::nQuality
```

Rendering quality (number of segments / rotation)

### 8.20.2.8 nVal

```
int16_t gslc_tsXGlowball::nVal
```

Current value.

### 8.20.2.9 nValLast

```
int16_t gslc_tsXGlowball::nValLast
```

Previous value.

### 8.20.2.10 pRings

```
gslc_tsXGlowballRing* gslc_tsXGlowball::pRings
```

Ring definition array.

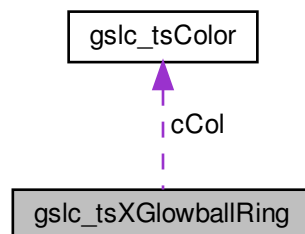
The documentation for this struct was generated from the following file:

- [src/elem/XGlowball.h](#)

## 8.21 gslc\_tsXGlowballRing Struct Reference

```
#include <XGlowball.h>
```

Collaboration diagram for `gslc_tsXGlowballRing`:



### Data Fields

- `uint8_t nRad1`
- `uint8_t nRad2`
- `gslc_tsColor cCol`

### 8.21.1 Field Documentation



## 8.21.1.1 cCol

```
gslc_tsColor gslc_tsXGlowballRing::cCol
```

## 8.21.1.2 nRad1

```
uint8_t gslc_tsXGlowballRing::nRad1
```

## 8.21.1.3 nRad2

```
uint8_t gslc_tsXGlowballRing::nRad2
```

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

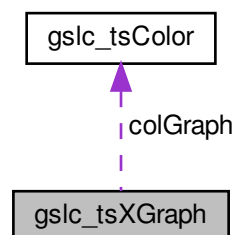
- [src/elem/XGlowball.h](#)

## 8.22 gslc\_tsXGraph Struct Reference

Extended data for Graph element.

```
#include <XGraph.h>
```

Collaboration diagram for gslc\_tsXGraph:



## Data Fields

- `int16_t * pBuf`  
*Ptr to the data buffer (circular buffer)*
- `uint8_t nMargin`  
*Margin for graph area within element rect.*
- `gslc_tsColor colGraph`  
*Color of the graph.*
- `gslc_teXGraphStyle eStyle`  
*Style of the graph.*
- `uint16_t nBufMax`  
*Maximum number of points in buffer.*
- `bool bScrollEn`  
*Enable for scrollbar.*
- `uint16_t nScrollPos`  
*Current scrollbar position.*
- `uint16_t nWndHeight`  
*Visible window height.*
- `uint16_t nWndWidth`  
*Visible window width.*
- `int16_t nPlotValMax`  
*Visible window maximum value.*
- `int16_t nPlotValMin`  
*Visible window minimum value.*
- `uint16_t nPlotIndMax`  
*Number of data points to show in window.*
- `uint16_t nBufCnt`  
*Number of points in buffer.*
- `uint16_t nPlotIndStart`  
*First row of current window.*

### 8.22.1 Detailed Description

Extended data for Graph element.

### 8.22.2 Field Documentation

#### 8.22.2.1 bScrollEn

```
bool gslc_tsXGraph::bScrollEn
```

Enable for scrollbar.

#### 8.22.2.2 colGraph

`gslc_tsColor` `gslc_tsXGraph::colGraph`

Color of the graph.

#### 8.22.2.3 eStyle

`gslc_teXGraphStyle` `gslc_tsXGraph::eStyle`

Style of the graph.

#### 8.22.2.4 nBufCnt

`uint16_t` `gslc_tsXGraph::nBufCnt`

Number of points in buffer.

#### 8.22.2.5 nBufMax

`uint16_t` `gslc_tsXGraph::nBufMax`

Maximum number of points in buffer.

#### 8.22.2.6 nMargin

`uint8_t` `gslc_tsXGraph::nMargin`

Margin for graph area within element rect.

#### 8.22.2.7 nPlotIndMax

`uint16_t` `gslc_tsXGraph::nPlotIndMax`

Number of data points to show in window.

**8.22.2.8 nPlotIndStart**

```
uint16_t gslc_tsXGraph::nPlotIndStart
```

First row of current window.

**8.22.2.9 nPlotValMax**

```
int16_t gslc_tsXGraph::nPlotValMax
```

Visible window maximum value.

**8.22.2.10 nPlotValMin**

```
int16_t gslc_tsXGraph::nPlotValMin
```

Visible window minimum value.

**8.22.2.11 nScrollPos**

```
uint16_t gslc_tsXGraph::nScrollPos
```

Current scrollbar position.

**8.22.2.12 nWndHeight**

```
uint16_t gslc_tsXGraph::nWndHeight
```

Visible window height.

**8.22.2.13 nWndWidth**

```
uint16_t gslc_tsXGraph::nWndWidth
```

Visible window width.



### 8.23.1 Detailed Description

Extended data for KeyPad element.

### 8.23.2 Field Documentation

#### 8.23.2.1 acBuffer

```
char gslc_tsXKeyPad::acBuffer[XKEYPAD_BUF_MAX]
```

Buffer storage for edit value string.

#### 8.23.2.2 nBufferLen

```
uint8_t gslc_tsXKeyPad::nBufferLen
```

Current number of characters stored in edit value string.

#### 8.23.2.3 nBufferMax

```
uint8_t gslc_tsXKeyPad::nBufferMax
```

Maximum number of characters stored in edit value string.

#### 8.23.2.4 nCursorPos

```
uint8_t gslc_tsXKeyPad::nCursorPos
```

Cursor position within the buffer.

#### 8.23.2.5 nFocusKeyInd

```
int16_t gslc_tsXKeyPad::nFocusKeyInd
```

Indicate key in focus (GSLC\_IND\_NONE if none)

#### 8.23.2.6 nGlowKeyInd

```
int16_t gslc_tsXKeyPad::nGlowKeyInd
```

Indicate key in glow (GSLC\_IND\_NONE if none)

#### 8.23.2.7 nScrollPos

```
uint8_t gslc_tsXKeyPad::nScrollPos
```

Display offset within the buffer.

#### 8.23.2.8 pConfig

```
gslc_tsXKeyPadCfg* gslc_tsXKeyPad::pConfig
```

Ptr to config struct (may be derived variant)

#### 8.23.2.9 pfuncCb

```
GSLC_CB_INPUT gslc_tsXKeyPad::pfuncCb
```

Callback function for KeyPad actions.

#### 8.23.2.10 pTargetRef

```
gslc_tsElemRef* gslc_tsXKeyPad::pTargetRef
```

Target element ref associated with keypad (GSLC\_CB\_INPUT)

#### 8.23.2.11 sRedraw

```
gslc_tsXKeyPadResult gslc_tsXKeyPad::sRedraw
```

Pending redraw state.

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

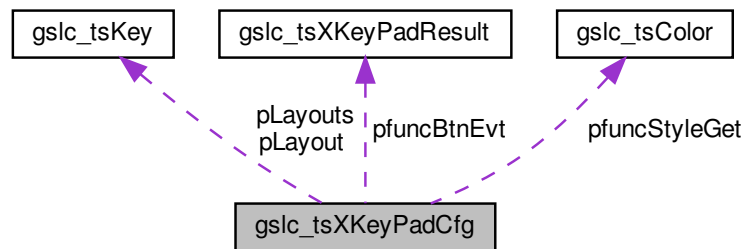
- [src/elem/XKeyPad.h](#)

## 8.24 gslc\_tsXKeyPadCfg Struct Reference

Configuration for the KeyPad.

```
#include <XKeyPad.h>
```

Collaboration diagram for gslc\_tsXKeyPadCfg:



### Data Fields

- `uint8_t nDispMax`  
*Maximum length to display.*
- `bool bRoundEn`  
*Enable rounded corners.*
- `int8_t nButtonSzW`  
*Button width (in pixels)*
- `int8_t nButtonSzH`  
*Button height (in pixels)*
- `int8_t nButtonSpaceX`  
*Button X spacing (in pixels)*
- `int8_t nButtonSpaceY`  
*Button Y spacing (in pixels)*
- `gslc_tsKey * pLayout`  
*Current selected layout.*
- `gslc_tsKey ** pLayouts`  
*Key Positions for each KeyPad layout.*
- `int8_t eLayoutDef`  
*Default KeyPad layout (type `gslc_teXKeyPadSel`)*
- `int8_t eLayoutSel`  
*Current KeyPad layout (type `gslc_teXKeyPadSel`)*
- `int16_t nFontId`  
*Configured font for KeyPad labels.*
- `int16_t nOffsetX`  
*Configured offset (X direction) for buttons from parent container.*
- `int16_t nOffsetY`  
*Configured offset (Y direction) for buttons from parent container.*



- [int8\\_t nFrameMargin](#)  
*Margin around text value field.*
- [uint8\\_t nMaxCols](#)  
*Maximum number of columns to occupy.*
- [uint8\\_t nMaxRows](#)  
*Maximum number of rows to occupy.*
- [GSLC\\_CB\\_XKEYPAD\\_RESET](#) [pfuncReset](#)  
*Callback function whenever keypad needs to be reset.*
- [GSLC\\_CB\\_XKEYPAD\\_TXT\\_INIT](#) [pfuncTxtInit](#)  
*Callback function whenever text string initialized.*
- [GSLC\\_CB\\_XKEYPAD\\_LABEL\\_GET](#) [pfuncLabelGet](#)  
*Callback function to get a key label.*
- [GSLC\\_CB\\_XKEYPAD\\_SYTLE\\_GET](#) [pfuncStyleGet](#)  
*Callback function to get a key's style.*
- [GSLC\\_CB\\_XKEYPAD\\_BTN\\_EVT](#) [pfuncBtnEvt](#)  
*Callback function to handle a key.*

### 8.24.1 Detailed Description

Configuration for the KeyPad.

### 8.24.2 Field Documentation

#### 8.24.2.1 bRoundEn

```
bool gslc_tsXKeyPadCfg::bRoundEn
```

Enable rounded corners.

#### 8.24.2.2 eLayoutDef

```
int8_t gslc_tsXKeyPadCfg::eLayoutDef
```

Default KeyPad layout (type [gslc\\_teXKeyPadSel](#))

#### 8.24.2.3 eLayoutSel

```
int8_t gslc_tsXKeyPadCfg::eLayoutSel
```

Current KeyPad layout (type [gslc\\_teXKeyPadSel](#))

#### 8.24.2.4 nButtonSpaceX

```
int8_t gslc_tsXKeyPadCfg::nButtonSpaceX
```

Button X spacing (in pixels)

#### 8.24.2.5 nButtonSpaceY

```
int8_t gslc_tsXKeyPadCfg::nButtonSpaceY
```

Button Y spacing (in pixels)

#### 8.24.2.6 nButtonSzH

```
int8_t gslc_tsXKeyPadCfg::nButtonSzH
```

Button height (in pixels)

#### 8.24.2.7 nButtonSzW

```
int8_t gslc_tsXKeyPadCfg::nButtonSzW
```

Button width (in pixels)

#### 8.24.2.8 nDispMax

```
uint8_t gslc_tsXKeyPadCfg::nDispMax
```

Maximum length to display.

#### 8.24.2.9 nFontId

```
int16_t gslc_tsXKeyPadCfg::nFontId
```

Configured font for KeyPad labels.

#### 8.24.2.10 nFrameMargin

```
int8_t gslc_tsXKeyPadCfg::nFrameMargin
```

Margin around text value field.

#### 8.24.2.11 nMaxCols

```
uint8_t gslc_tsXKeyPadCfg::nMaxCols
```

Maximum number of columns to occupy.

#### 8.24.2.12 nMaxRows

```
uint8_t gslc_tsXKeyPadCfg::nMaxRows
```

Maximum number of rows to occupy.

#### 8.24.2.13 nOffsetX

```
int16_t gslc_tsXKeyPadCfg::nOffsetX
```

Configured offset (X direction) for buttons from parent container.

#### 8.24.2.14 nOffsetY

```
int16_t gslc_tsXKeyPadCfg::nOffsetY
```

Configured offset (Y direction) for buttons from parent container.

#### 8.24.2.15 pfuncBtnEvt

```
GSLC_CB_XKEYPAD_BTN_EVT gslc_tsXKeyPadCfg::pfuncBtnEvt
```

Callback function to handle a key.

#### 8.24.2.16 pfuncLabelGet

[GSLC\\_CB\\_XKEYPAD\\_LABEL\\_GET](#) `gslc_tsXKeyPadCfg::pfuncLabelGet`

Callback function to get a key label.

#### 8.24.2.17 pfuncReset

[GSLC\\_CB\\_XKEYPAD\\_RESET](#) `gslc_tsXKeyPadCfg::pfuncReset`

Callback function whenever keypad needs to be reset.

#### 8.24.2.18 pfuncStyleGet

[GSLC\\_CB\\_XKEYPAD\\_SYTLE\\_GET](#) `gslc_tsXKeyPadCfg::pfuncStyleGet`

Callback function to get a key's style.

#### 8.24.2.19 pfuncTxtInit

[GSLC\\_CB\\_XKEYPAD\\_TXT\\_INIT](#) `gslc_tsXKeyPadCfg::pfuncTxtInit`

Callback function whenever text string initialized.

#### 8.24.2.20 pLayout

[gslc\\_tsKey\\*](#) `gslc_tsXKeyPadCfg::pLayout`

Current selected layout.

#### 8.24.2.21 pLayouts

[gslc\\_tsKey\\*\\*](#) `gslc_tsXKeyPadCfg::pLayouts`

Key Positions for each KeyPad layout.

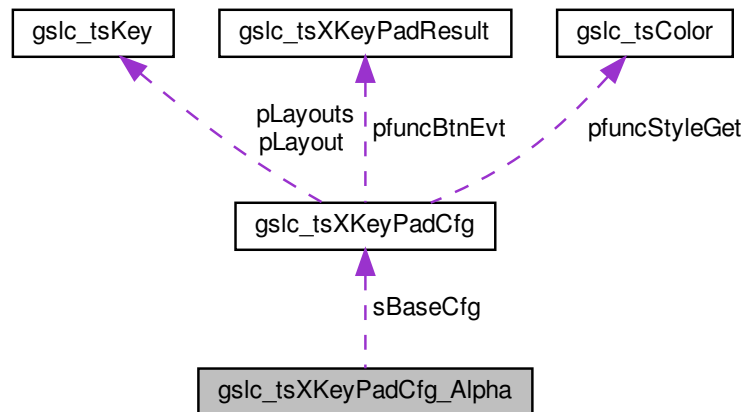
The documentation for this struct was generated from the following file:

- [src/elem/XKeyPad.h](#)

## 8.25 gslc\_tsXKeyPadCfg\_Alpha Struct Reference

```
#include <XKeyPad_Alpha.h>
```

Collaboration diagram for gslc\_tsXKeyPadCfg\_Alpha:



### Data Fields

- [gslc\\_tsXKeyPadCfg sBaseCfg](#)  
*KeyPad base config struct.*

### 8.25.1 Field Documentation

#### 8.25.1.1 sBaseCfg

```
gslc\_tsXKeyPadCfg gslc_tsXKeyPadCfg_Alpha::sBaseCfg
```

KeyPad base config struct.

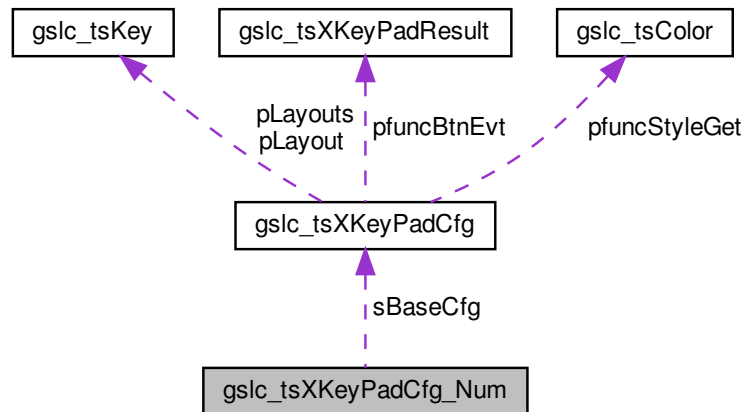
The documentation for this struct was generated from the following file:

- [src/elem/XKeyPad\\_Alpha.h](#)

## 8.26 gslc\_tsXKeyPadCfg\_Num Struct Reference

```
#include <XKeyPad_Num.h>
```

Collaboration diagram for gslc\_tsXKeyPadCfg\_Num:



### Data Fields

- [gslc\\_tsXKeyPadCfg sBaseCfg](#)  
*KeyPad base config struct.*
- [bool bFloatEn](#)  
*Enable floating point (ie. decimal point)*
- [bool bSignEn](#)  
*Enable negative numbers.*
- [bool bValPositive](#)  
*Is the current value positive? (1=positive, 0=negative)*
- [bool bValDecimalPt](#)  
*Does the current value include a decimal point?*

### 8.26.1 Field Documentation

#### 8.26.1.1 bFloatEn

```
bool gslc_tsXKeyPadCfg_Num::bFloatEn
```

Enable floating point (ie. decimal point)

## 8.26.1.2 bSignEn

```
bool gslc_tsXKeyPadCfg_Num::bSignEn
```

Enable negative numbers.

## 8.26.1.3 bValDecimalPt

```
bool gslc_tsXKeyPadCfg_Num::bValDecimalPt
```

Does the current value include a decimal point?

## 8.26.1.4 bValPositive

```
bool gslc_tsXKeyPadCfg_Num::bValPositive
```

Is the current value positive? (1=positive, 0=negative)

## 8.26.1.5 sBaseCfg

```
gslc_tsXKeyPadCfg gslc_tsXKeyPadCfg_Num::sBaseCfg
```

KeyPad base config struct.

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

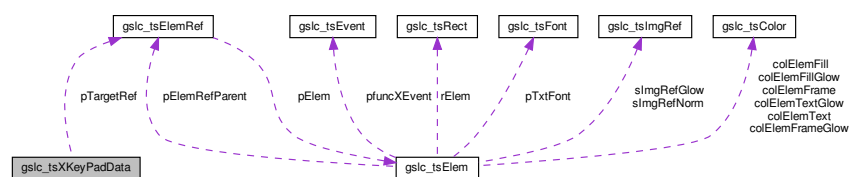
- [src/elem/XKeyPad\\_Num.h](#)

## 8.27 gslc\_tsXKeyPadData Struct Reference

Input callback data structure.

```
#include <XKeyPad.h>
```

Collaboration diagram for gslc\_tsXKeyPadData:



## Data Fields

- `char * pStr`  
*Final value of edited value field.*
- `gslc_tsElemRef * pTargetRef`  
*Target element reference to receive the value.*

### 8.27.1 Detailed Description

Input callback data structure.

- This struct is returned in `GSLC_CB_INPUT` when the KeyPad edits are complete, and is used to provide the resulting edited value.

### 8.27.2 Field Documentation

#### 8.27.2.1 pStr

```
char* gslc_tsXKeyPadData::pStr
```

Final value of edited value field.

#### 8.27.2.2 pTargetRef

```
gslc_tsElemRef* gslc_tsXKeyPadData::pTargetRef
```

Target element reference to receive the value.

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

- `src/elem/XKeyPad.h`

## 8.28 gslc\_tsXKeyPadResult Struct Reference

Return status for XKeyPad.

```
#include <XKeyPad.h>
```



## Data Fields

- `int16_t eRedrawState`  
*XKeyPad pending redraw state.*
- `int16_t nRedrawKeyId1`  
*XKeyPad specific key (#1) to redraw (-1 for none)*
- `int16_t nRedrawKeyId2`  
*XKeyPad specific key (#2) to redraw (-1 for none)*

### 8.28.1 Detailed Description

Return status for XKeyPad.

- Includes any pending redraw state

### 8.28.2 Field Documentation

#### 8.28.2.1 eRedrawState

```
int16_t gslc_tsXKeyPadResult::eRedrawState
```

XKeyPad pending redraw state.

#### 8.28.2.2 nRedrawKeyId1

```
int16_t gslc_tsXKeyPadResult::nRedrawKeyId1
```

XKeyPad specific key (#1) to redraw (-1 for none)

#### 8.28.2.3 nRedrawKeyId2

```
int16_t gslc_tsXKeyPadResult::nRedrawKeyId2
```

XKeyPad specific key (#2) to redraw (-1 for none)

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

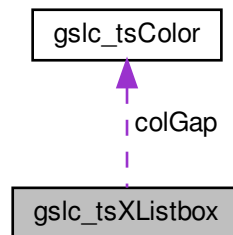
- `src/elem/XKeyPad.h`

## 8.29 gslc\_tsXListbox Struct Reference

Extended data for Listbox element.

```
#include <XListbox.h>
```

Collaboration diagram for gslc\_tsXListbox:



### Data Fields

- uint8\_t \* [pBufItems](#)  
*Buffer containing items.*
- uint16\_t [nBufItemsMax](#)  
*Max size of buffer containing items.*
- uint16\_t [nBufItemsPos](#)  
*Current buffer position.*
- int16\_t [nItemCnt](#)  
*Number of items in the list.*
- int8\_t [nCols](#)  
*Number of columns.*
- int8\_t [nRows](#)  
*Number of columns (or XLSITBOX\_SIZE\_AUTO to calculate)*
- bool [bNeedRecalc](#)  
*Determine if sizing may need recalc.*
- int8\_t [nMarginW](#)  
*Margin inside main listbox area (X offset)*
- int8\_t [nMarginH](#)  
*Margin inside main listbox area (Y offset)*
- int16\_t [nItemW](#)  
*Width of listbox item.*
- int16\_t [nItemH](#)  
*Height of listbox item.*
- int8\_t [nItemGap](#)  
*Gap between listbox items.*
- [gslc\\_tsColor](#) [colGap](#)  
*Gap color.*

- bool [bltemAutoSizeW](#)  
*Enable auto-sizing of items (in width)*
- bool [bltemAutoSizeH](#)  
*Enable auto-sizing of items (in height)*
- int16\_t [nltemCurSel](#)  
*Currently selected item (XLISTBOX\_SEL\_NONE for none)*
- int16\_t [nltemCurSelLast](#)  
*Old selected item to redraw (XLISTBOX\_SEL\_NONE for none)*
- int16\_t [nltemSavedSel](#)  
*Persistent selected item (ie. saved selection)*
- int16\_t [nltemTop](#)  
*Item to show at top of list after scrolling (0 is default)*
- bool [bGlowLast](#)  
*Last glow state.*
- bool [bFocusLast](#)  
*Last focus state // TODO: Merge with bGlowLast.*
- [GSLC\\_CB\\_XLISTBOX\\_SEL](#) [pfuncXSel](#)  
*Callback func ptr for selection update.*

### 8.29.1 Detailed Description

Extended data for Listbox element.

### 8.29.2 Field Documentation

#### 8.29.2.1 bFocusLast

```
bool gslc_tsXListbox::bFocusLast
```

Last focus state // TODO: Merge with bGlowLast.

#### 8.29.2.2 bGlowLast

```
bool gslc_tsXListbox::bGlowLast
```

Last glow state.

#### 8.29.2.3 bltemAutoSizeH

```
bool gslc_tsXListbox::bltemAutoSizeH
```

Enable auto-sizing of items (in height)

#### 8.29.2.4 bItemAutoSizeW

```
bool gslc_tsXListbox::bItemAutoSizeW
```

Enable auto-sizing of items (in width)

#### 8.29.2.5 bNeedRecalc

```
bool gslc_tsXListbox::bNeedRecalc
```

Determine if sizing may need recalc.

#### 8.29.2.6 colGap

```
gslc_tsColor gslc_tsXListbox::colGap
```

Gap color.

#### 8.29.2.7 nBufItemsMax

```
uint16_t gslc_tsXListbox::nBufItemsMax
```

Max size of buffer containing items.

#### 8.29.2.8 nBufItemsPos

```
uint16_t gslc_tsXListbox::nBufItemsPos
```

Current buffer position.

#### 8.29.2.9 nCols

```
int8_t gslc_tsXListbox::nCols
```

Number of columns.

**8.29.2.10 nItemCnt**

```
int16_t gslc_tsXListbox::nItemCnt
```

Number of items in the list.

**8.29.2.11 nItemCurSel**

```
int16_t gslc_tsXListbox::nItemCurSel
```

Currently selected item (XLISTBOX\_SEL\_NONE for none)

**8.29.2.12 nItemCurSelLast**

```
int16_t gslc_tsXListbox::nItemCurSelLast
```

Old selected item to redraw (XLISTBOX\_SEL\_NONE for none)

**8.29.2.13 nItemGap**

```
int8_t gslc_tsXListbox::nItemGap
```

Gap between listbox items.

**8.29.2.14 nItemH**

```
int16_t gslc_tsXListbox::nItemH
```

Height of listbox item.

**8.29.2.15 nItemSavedSel**

```
int16_t gslc_tsXListbox::nItemSavedSel
```

Persistent selected item (ie. saved selection)

**8.29.2.16 nItemTop**

```
int16_t gslc_tsXListBox::nItemTop
```

Item to show at top of list after scrolling (0 is default)

**8.29.2.17 nItemW**

```
int16_t gslc_tsXListBox::nItemW
```

Width of listbox item.

**8.29.2.18 nMarginH**

```
int8_t gslc_tsXListBox::nMarginH
```

Margin inside main listbox area (Y offset)

**8.29.2.19 nMarginW**

```
int8_t gslc_tsXListBox::nMarginW
```

Margin inside main listbox area (X offset)

**8.29.2.20 nRows**

```
int8_t gslc_tsXListBox::nRows
```

Number of columns (or XLSITBOX\_SIZE\_AUTO to calculate)

**8.29.2.21 pBufItems**

```
uint8_t* gslc_tsXListBox::pBufItems
```

Buffer containing items.

## 8.29.2.22 pfuncXSel

`GSLC_CB_XLISTBOX_SEL` `gslc_tsXListbox::pfuncXSel`

Callback func ptr for selection update.

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

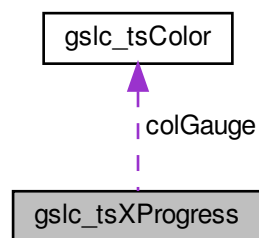
- `src/elem/XListbox.h`

## 8.30 gslc\_tsXProgress Struct Reference

Extended data for Gauge element.

```
#include <XProgress.h>
```

Collaboration diagram for `gslc_tsXProgress`:



## Data Fields

- `int16_t nMin`  
*Minimum control value.*
- `int16_t nMax`  
*Maximum control value.*
- `int16_t nVal`  
*Current control value.*
- `int16_t nValLast`  
*Last value.*
- `bool bValLastValid`  
*Last value valid?*
- `gslc_tsColor colGauge`  
*Color of gauge fill bar.*
- `bool bVert`  
*Vertical if true, else Horizontal.*
- `bool bFlip`  
*Reverse direction of gauge.*

### 8.30.1 Detailed Description

Extended data for Gauge element.

### 8.30.2 Field Documentation

#### 8.30.2.1 bFlip

```
bool gslc_tsXProgress::bFlip
```

Reverse direction of gauge.

#### 8.30.2.2 bValLastValid

```
bool gslc_tsXProgress::bValLastValid
```

Last value valid?

#### 8.30.2.3 bVert

```
bool gslc_tsXProgress::bVert
```

Vertical if true, else Horizontal.

#### 8.30.2.4 colGauge

```
gslc_tsColor gslc_tsXProgress::colGauge
```

Color of gauge fill bar.

#### 8.30.2.5 nMax

```
int16_t gslc_tsXProgress::nMax
```

Maximum control value.



#### 8.30.2.6 nMin

```
int16_t gslc_tsXProgress::nMin
```

Minimum control value.

#### 8.30.2.7 nVal

```
int16_t gslc_tsXProgress::nVal
```

Current control value.

#### 8.30.2.8 nValLast

```
int16_t gslc_tsXProgress::nValLast
```

Last value.

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

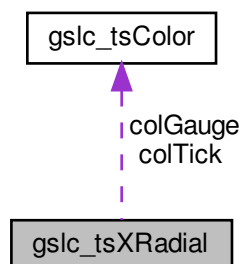
- [src/elem/XProgress.h](#)

## 8.31 gslc\_tsXRadial Struct Reference

Extended data for Gauge element.

```
#include <XRadial.h>
```

Collaboration diagram for gslc\_tsXRadial:



## Data Fields

- `int16_t nMin`  
*Minimum control value.*
- `int16_t nMax`  
*Maximum control value.*
- `int16_t nVal`  
*Current control value.*
- `int16_t nValLast`  
*Last value.*
- `bool bValLastValid`  
*Last value valid?*
- `gslc_tsColor colGauge`  
*Color of gauge fill bar.*
- `gslc_tsColor colTick`  
*Color of gauge tick marks.*
- `uint16_t nTickCnt`  
*Number of gauge tick marks.*
- `uint16_t nTickLen`  
*Length of gauge tick marks.*
- `bool bFlip`  
*Reverse direction of gauge.*
- `uint16_t nIndicLen`  
*Indicator length.*
- `uint16_t nIndicTip`  
*Size of tip at end of indicator.*
- `bool bIndicFill`  
*Fill the indicator if true.*

### 8.31.1 Detailed Description

Extended data for Gauge element.

### 8.31.2 Field Documentation

#### 8.31.2.1 bFlip

```
bool gslc_tsXRadial::bFlip
```

Reverse direction of gauge.

#### 8.31.2.2 bIndicFill

```
bool gslc_tsXRadial::bIndicFill
```

Fill the indicator if true.

#### 8.31.2.3 bValLastValid

```
bool gslc_tsXRadial::bValLastValid
```

Last value valid?

#### 8.31.2.4 colGauge

```
gslc_tsColor gslc_tsXRadial::colGauge
```

Color of gauge fill bar.

#### 8.31.2.5 colTick

```
gslc_tsColor gslc_tsXRadial::colTick
```

Color of gauge tick marks.

#### 8.31.2.6 nIndicLen

```
uint16_t gslc_tsXRadial::nIndicLen
```

Indicator length.

#### 8.31.2.7 nIndicTip

```
uint16_t gslc_tsXRadial::nIndicTip
```

Size of tip at end of indicator.

**8.31.2.8 nMax**

```
int16_t gslc_tsXRadial::nMax
```

Maximum control value.

**8.31.2.9 nMin**

```
int16_t gslc_tsXRadial::nMin
```

Minimum control value.

**8.31.2.10 nTickCnt**

```
uint16_t gslc_tsXRadial::nTickCnt
```

Number of gauge tick marks.

**8.31.2.11 nTickLen**

```
uint16_t gslc_tsXRadial::nTickLen
```

Length of gauge tick marks.

**8.31.2.12 nVal**

```
int16_t gslc_tsXRadial::nVal
```

Current control value.

**8.31.2.13 nValLast**

```
int16_t gslc_tsXRadial::nValLast
```

Last value.

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

- [src/ele/XRadial.h](#)

## 8.32 gslc\_tsXRamp Struct Reference

Extended data for Gauge element.

```
#include <XRamp.h>
```

### Data Fields

- `int16_t nMin`  
*Minimum control value.*
- `int16_t nMax`  
*Maximum control value.*
- `int16_t nVal`  
*Current control value.*
- `int16_t nValLast`  
*Last value.*
- `bool bValLastValid`  
*Last value valid?*

### 8.32.1 Detailed Description

Extended data for Gauge element.

### 8.32.2 Field Documentation

#### 8.32.2.1 bValLastValid

```
bool gslc_tsXRamp::bValLastValid
```

Last value valid?

#### 8.32.2.2 nMax

```
int16_t gslc_tsXRamp::nMax
```

Maximum control value.

#### 8.32.2.3 nMin

```
int16_t gslc_tsXRamp::nMin
```

Minimum control value.

#### 8.32.2.4 nVal

```
int16_t gslc_tsXRamp::nVal
```

Current control value.

#### 8.32.2.5 nValLast

```
int16_t gslc_tsXRamp::nValLast
```

Last value.

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

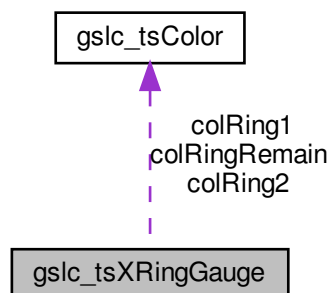
- [src/elem/XRamp.h](#)

## 8.33 gslc\_tsXRingGauge Struct Reference

Extended data for XRingGauge element.

```
#include <XRingGauge.h>
```

Collaboration diagram for gslc\_tsXRingGauge:



## Data Fields

- int16\_t [nValMin](#)
- int16\_t [nValMax](#)
- int16\_t [nAngStart](#)
- int16\_t [nAngRange](#)
- int16\_t [nQuality](#)
- int8\_t [nThickness](#)
- bool [bGradient](#)
- uint8\_t [nSegGap](#)
- [gslc\\_tsColor](#) [colRing1](#)
- [gslc\\_tsColor](#) [colRing2](#)
- [gslc\\_tsColor](#) [colRingRemain](#)
- int16\_t [nVal](#)  
*Current position value.*
- int16\_t [nValLast](#)  
*Previous position value.*
- char [acStrLast](#) [[XRING\\_STR\\_MAX](#)]

### 8.33.1 Detailed Description

Extended data for XRingGauge element.

### 8.33.2 Field Documentation

#### 8.33.2.1 [acStrLast](#)

```
char gslc_tsXRingGauge::acStrLast [XRING\_STR\_MAX]
```

#### 8.33.2.2 [bGradient](#)

```
bool gslc_tsXRingGauge::bGradient
```

#### 8.33.2.3 [colRing1](#)

```
gslc\_tsColor gslc_tsXRingGauge::colRing1
```

#### 8.33.2.4 colRing2

`gslc_tsColor` `gslc_tsXRingGauge::colRing2`

#### 8.33.2.5 colRingRemain

`gslc_tsColor` `gslc_tsXRingGauge::colRingRemain`

#### 8.33.2.6 nAngRange

`int16_t` `gslc_tsXRingGauge::nAngRange`

#### 8.33.2.7 nAngStart

`int16_t` `gslc_tsXRingGauge::nAngStart`

#### 8.33.2.8 nQuality

`int16_t` `gslc_tsXRingGauge::nQuality`

#### 8.33.2.9 nSegGap

`uint8_t` `gslc_tsXRingGauge::nSegGap`

#### 8.33.2.10 nThickness

`int8_t` `gslc_tsXRingGauge::nThickness`



#### 8.33.2.11 nVal

```
int16_t gslc_tsXRingGauge::nVal
```

Current position value.

#### 8.33.2.12 nValLast

```
int16_t gslc_tsXRingGauge::nValLast
```

Previous position value.

#### 8.33.2.13 nValMax

```
int16_t gslc_tsXRingGauge::nValMax
```

#### 8.33.2.14 nValMin

```
int16_t gslc_tsXRingGauge::nValMin
```

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

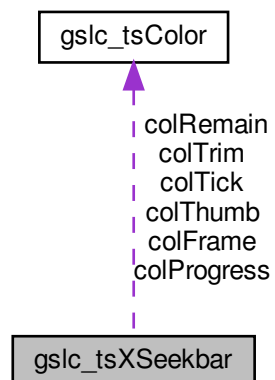
- [src/elm/XRingGauge.h](#)

## 8.34 gslc\_tsXSeekBar Struct Reference

Extended data for Seekbar element.

```
#include <XSeekBar.h>
```

Collaboration diagram for gslc\_tsXSeekBar:



## Data Fields

- bool [bVert](#)  
*Orientation: true if vertical, else horizontal.*
- uint8\_t [nProgressW](#)  
*Width of progress track.*
- uint8\_t [nRemainW](#)  
*Width of remaining track.*
- uint8\_t [nThumbSz](#)  
*Size of the thumb control.*
- int16\_t [nPosMin](#)  
*Minimum position value of the slider.*
- int16\_t [nPosMax](#)  
*Maximum position value of the slider.*
- [gslc\\_tsColor](#) [colProgress](#)  
*Style: color of progress fill bar.*
- [gslc\\_tsColor](#) [colRemain](#)  
*Style: color remaining fill bar.*
- [gslc\\_tsColor](#) [colThumb](#)  
*Style: color of thumb.*
- uint16\_t [nTickDiv](#)  
*Style: number of tickmark divisions (0 for none)*
- int16\_t [nTickLen](#)  
*Style: length of tickmarks.*
- [gslc\\_tsColor](#) [colTick](#)  
*Style: color of ticks.*
- bool [bTrimThumb](#)  
*Style: show a trim color for thumb.*
- [gslc\\_tsColor](#) [colTrim](#)  
*Style: color of trim.*
- bool [bFrameThumb](#)  
*Style: draw frame around thumb.*
- [gslc\\_tsColor](#) [colFrame](#)  
*Style: color of trim.*
- int16\_t [nPos](#)  
*Current position value of the slider.*
- [GSLC\\_CB\\_XSEEKBAR\\_POS](#) [pfuncXPos](#)  
*Callback func ptr for position update.*

### 8.34.1 Detailed Description

Extended data for Seekbar element.

### 8.34.2 Field Documentation

#### 8.34.2.1 bFrameThumb

```
bool gslc_tsXSeekBar::bFrameThumb
```

Style: draw frame around thumb.

#### 8.34.2.2 bTrimThumb

```
bool gslc_tsXSeekBar::bTrimThumb
```

Style: show a trim color for thumb.

#### 8.34.2.3 bVert

```
bool gslc_tsXSeekBar::bVert
```

Orientation: true if vertical, else horizontal.

#### 8.34.2.4 colFrame

```
gslc_tsColor gslc_tsXSeekBar::colFrame
```

Style: color of trim.

#### 8.34.2.5 colProgress

```
gslc_tsColor gslc_tsXSeekBar::colProgress
```

Style: color of progress fill bar.

#### 8.34.2.6 colRemain

```
gslc_tsColor gslc_tsXSeekBar::colRemain
```

Style: color remaining fill bar.

#### 8.34.2.7 colThumb

`gslc_tsColor` `gslc_tsXSeekBar::colThumb`

Style: color of thumb.

#### 8.34.2.8 colTick

`gslc_tsColor` `gslc_tsXSeekBar::colTick`

Style: color of ticks.

#### 8.34.2.9 colTrim

`gslc_tsColor` `gslc_tsXSeekBar::colTrim`

Style: color of trim.

#### 8.34.2.10 nPos

`int16_t` `gslc_tsXSeekBar::nPos`

Current position value of the slider.

#### 8.34.2.11 nPosMax

`int16_t` `gslc_tsXSeekBar::nPosMax`

Maximum position value of the slider.

#### 8.34.2.12 nPosMin

`int16_t` `gslc_tsXSeekBar::nPosMin`

Minimum position value of the slider.

**8.34.2.13 nProgressW**

```
uint8_t gslc_tsXSeekBar::nProgressW
```

Width of progress track.

**8.34.2.14 nRemainW**

```
uint8_t gslc_tsXSeekBar::nRemainW
```

Width of remaining track.

**8.34.2.15 nThumbSz**

```
uint8_t gslc_tsXSeekBar::nThumbSz
```

Size of the thumb control.

**8.34.2.16 nTickDiv**

```
uint16_t gslc_tsXSeekBar::nTickDiv
```

Style: number of tickmark divisions (0 for none)

**8.34.2.17 nTickLen**

```
int16_t gslc_tsXSeekBar::nTickLen
```

Style: length of tickmarks.

**8.34.2.18 pfuncXPos**

```
GSLC\_CB\_XSEEKBAR\_POS gslc_tsXSeekBar::pfuncXPos
```

Callback func ptr for position update.

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

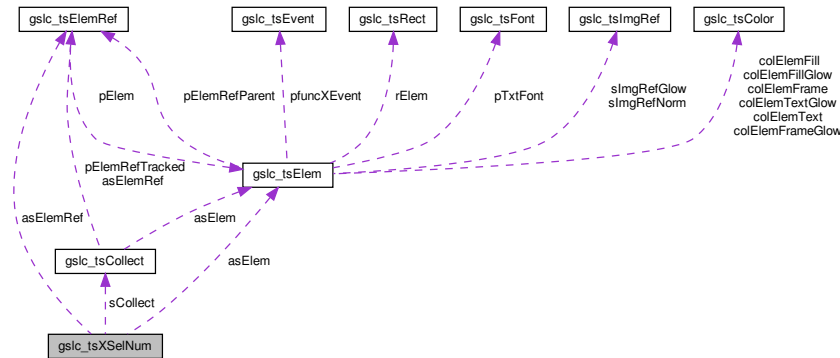
- [src/elem/XSeekBar.h](#)

## 8.35 gslc\_tsXSelNum Struct Reference

Extended data for SelNum element.

```
#include <XSelNum.h>
```

Collaboration diagram for gslc\_tsXSelNum:



### Data Fields

- `int16_t nCounter`  
*Counter for demo purposes.*
- `gslc_tsCollect sCollect`  
*Collection management for sub-elements.*
- `gslc_tsElemRef asElemRef` [4]  
*Storage for sub-element references.*
- `gslc_tsElem asElem` [4]  
*Storage for sub-elements.*
- `char acElemTxt` [4][`SELNUM_STR_LEN`]  
*Storage for strings.*

### 8.35.1 Detailed Description

Extended data for SelNum element.

### 8.35.2 Field Documentation

#### 8.35.2.1 acElemTxt

```
char gslc_tsXSelNum::acElemTxt[4][SELNUM_STR_LEN]
```

Storage for strings.

### 8.35.2.2 asElem

```
gslc_tsElem gslc_tsXSelNum::asElem[4]
```

Storage for sub-elements.

### 8.35.2.3 asElemRef

```
gslc_tsElemRef gslc_tsXSelNum::asElemRef[4]
```

Storage for sub-element references.

### 8.35.2.4 nCounter

```
int16_t gslc_tsXSelNum::nCounter
```

Counter for demo purposes.

### 8.35.2.5 sCollect

```
gslc_tsCollect gslc_tsXSelNum::sCollect
```

Collection management for sub-elements.

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

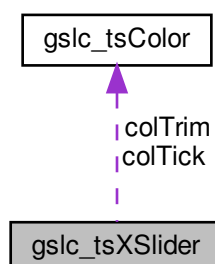
- [src/elm/XSelNum.h](#)

## 8.36 gslc\_tsXSlider Struct Reference

Extended data for Slider element.

```
#include <XSlider.h>
```

Collaboration diagram for gslc\_tsXSlider:



## Data Fields

- bool [bVert](#)  
*Orientation: true if vertical, else horizontal.*
- bool [bSnapEn](#)  
*Enable for touch snap behavior.*
- int16\_t [nThumbSz](#)  
*Size of the thumb control.*
- int16\_t [nPosMin](#)  
*Minimum position value of the slider.*
- int16\_t [nPosMax](#)  
*Maximum position value of the slider.*
- uint16\_t [nTickDiv](#)  
*Style: number of tickmark divisions (0 for none)*
- int16\_t [nTickLen](#)  
*Style: length of tickmarks.*
- [gslc\\_tsColor](#) [colTick](#)  
*Style: color of ticks.*
- bool [bTrim](#)  
*Style: show a trim color.*
- [gslc\\_tsColor](#) [colTrim](#)  
*Style: color of trim.*
- int16\_t [nPos](#)  
*Current position value of the slider.*
- [GSLC\\_CB\\_XSLIDER\\_POS](#) [pfuncXPos](#)  
*Callback func ptr for position update.*

### 8.36.1 Detailed Description

Extended data for Slider element.

### 8.36.2 Field Documentation

#### 8.36.2.1 bSnapEn

```
bool gslc_tsXSlider::bSnapEn
```

Enable for touch snap behavior.

#### 8.36.2.2 bTrim

```
bool gslc_tsXSlider::bTrim
```

Style: show a trim color.



### 8.36.2.3 bVert

```
bool gslc_tsXSlider::bVert
```

Orientation: true if vertical, else horizontal.

### 8.36.2.4 colTick

```
gslc_tsColor gslc_tsXSlider::colTick
```

Style: color of ticks.

### 8.36.2.5 colTrim

```
gslc_tsColor gslc_tsXSlider::colTrim
```

Style: color of trim.

### 8.36.2.6 nPos

```
int16_t gslc_tsXSlider::npos
```

Current position value of the slider.

### 8.36.2.7 nPosMax

```
int16_t gslc_tsXSlider::nposMax
```

Maximum position value of the slider.

### 8.36.2.8 nPosMin

```
int16_t gslc_tsXSlider::nposMin
```

Minimum position value of the slider.

### 8.36.2.9 nThumbSz

```
int16_t gslc_tsXSlider::nThumbSz
```

Size of the thumb control.

### 8.36.2.10 nTickDiv

```
uint16_t gslc_tsXSlider::nTickDiv
```

Style: number of tickmark divisions (0 for none)

### 8.36.2.11 nTickLen

```
int16_t gslc_tsXSlider::nTickLen
```

Style: length of tickmarks.

### 8.36.2.12 pfuncXPos

```
GSLC_CB_XSLIDER_POS gslc_tsXSlider::pfuncXPos
```

Callback func ptr for position update.

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

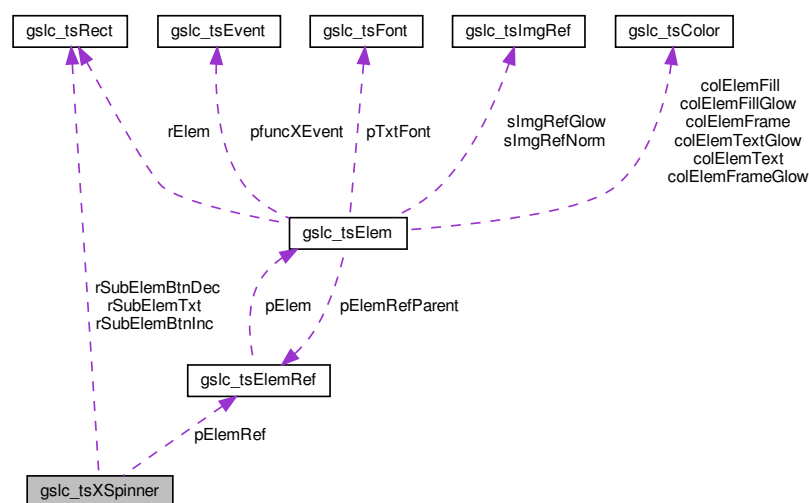
- [src/elem/XSlider.h](#)

## 8.37 gslc\_tsXSpinner Struct Reference

Extended data for Spinner element.

```
#include <XSpinner.h>
```

Collaboration diagram for `gslc_tsXSpinner`:



## Data Fields

- `int16_t nMin`  
*Minimum control value.*
- `int16_t nMax`  
*Maximum control value.*
- `int16_t nIncr`  
*Increment by value.*
- `int16_t nCounter`  
*Current value.*
- `GSLC_CB_INPUT pfuncXInput`  
*Callback func ptr for input ready.*
- `gslc_tsElemRef * pElemRef`  
*Save our ElemRef for the callback.*
- `int16_t nFontId`  
*Configured font for Spinner.*
- `gslc_tsRect rSubElemTxt`  
*Our virtual Text Field.*
- `gslc_tsRect rSubElemBtnInc`  
*Our virtual Increment Text Button.*
- `gslc_tsRect rSubElemBtnDec`  
*Our virtual Decrement Text Button.*
- `char acElemTxt [1][XSPINNER_STR_LEN]`  
*Storage for strings.*
- `char acIncr [2]`  
*Increment character string.*
- `char acDecr [2]`  
*Decrement character string.*

### 8.37.1 Detailed Description

Extended data for Spinner element.

### 8.37.2 Field Documentation

#### 8.37.2.1 acDecr

```
char gslc_tsXSpinner::acDecr[2]
```

Decrement character string.

#### 8.37.2.2 acElemTxt

```
char gslc_tsXSpinner::acElemTxt[1][XSPINNER_STR_LEN]
```

Storage for strings.

#### 8.37.2.3 acIncr

```
char gslc_tsXSpinner::acIncr[2]
```

Increment character string.

#### 8.37.2.4 nCounter

```
int16_t gslc_tsXSpinner::nCounter
```

Current value.

#### 8.37.2.5 nFontId

```
int16_t gslc_tsXSpinner::nFontId
```

Configured font for Spinner.

#### 8.37.2.6 nIncr

```
int16_t gslc_tsXSpinner::nIncr
```

Increment by value.

#### 8.37.2.7 nMax

```
int16_t gslc_tsXSpinner::nMax
```

Maximum control value.

#### 8.37.2.8 nMin

```
int16_t gslc_tsXSpinner::nMin
```

Minimum control value.

#### 8.37.2.9 pElemRef

```
gslc_tsElemRef* gslc_tsXSpinner::pElemRef
```

Save our ElemRef for the callback.

#### 8.37.2.10 pfuncXInput

```
GSLC_CB_INPUT gslc_tsXSpinner::pfuncXInput
```

Callback func ptr for input ready.

#### 8.37.2.11 rSubElemBtnDec

```
gslc_tsRect gslc_tsXSpinner::rSubElemBtnDec
```

Our virtual Decrement Text Button.

#### 8.37.2.12 rSubElemBtnInc

```
gslc_tsRect gslc_tsXSpinner::rSubElemBtnInc
```

Our virtual Increment Text Button.

#### 8.37.2.13 rSubElemTxt

```
gslc_tsRect gslc_tsXSpinner::rSubElemTxt
```

Our virtual Text Field.

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

- [src/ele/XSpinner.h](#)

## 8.38 gslc\_tsXTemplate Struct Reference

Callback function for slider feedback.

```
#include <XTemplate.h>
```

### 8.38.1 Detailed Description

Callback function for slider feedback.

Extended data for Slider element

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

- [src/elem/XTemplate.h](#)

## 8.39 gslc\_tsXTextbox Struct Reference

Extended data for Textbox element.

```
#include <XTextbox.h>
```

### Data Fields

- `char * pBuf`  
*Ptr to the text buffer (circular buffer)*
- `int8_t nMarginX`  
*Margin for text area within element rect (X)*
- `int8_t nMarginY`  
*Margin for text area within element rect (Y)*
- `bool bWrapEn`  
*Enable for line wrapping.*
- `uint16_t nBufRows`  
*Number of rows in buffer.*
- `uint16_t nBufCols`  
*Number of columns in buffer.*
- `bool bScrollEn`  
*Enable for scrollbar.*
- `uint16_t nScrollPos`  
*Current scrollbar position.*
- `uint8_t nChSizeX`  
*Width of characters (pixels)*
- `uint8_t nChSizeY`  
*Height of characters (pixels)*
- `uint8_t nWndCols`  
*Window X size.*
- `uint8_t nWndRows`

- Window Y size.*
- uint8\_t [nCurPosX](#)
- Cursor X position.*
- uint8\_t [nCurPosY](#)
- Cursor Y position.*
- uint8\_t [nBufPosX](#)
- Buffer X position.*
- uint8\_t [nBufPosY](#)
- Buffer Y position.*
- uint8\_t [nWndRowStart](#)
- First row of current window.*
- int16\_t [nRedrawRow](#)
- Specific row to update in redraw (if not -1)*

### 8.39.1 Detailed Description

Extended data for Textbox element.

### 8.39.2 Field Documentation

#### 8.39.2.1 bScrollEn

```
bool gslc_tsXTextbox::bScrollEn
```

Enable for scrollbar.

#### 8.39.2.2 bWrapEn

```
bool gslc_tsXTextbox::bWrapEn
```

Enable for line wrapping.

#### 8.39.2.3 nBufCols

```
uint16_t gslc_tsXTextbox::nBufCols
```

Number of columns in buffer.

#### 8.39.2.4 nBufPosX

```
uint8_t gslc_tsXTextbox::nBufPosX
```

Buffer X position.

#### 8.39.2.5 nBufPosY

```
uint8_t gslc_tsXTextbox::nBufPosY
```

Buffer Y position.

#### 8.39.2.6 nBufRows

```
uint16_t gslc_tsXTextbox::nBufRows
```

Number of rows in buffer.

#### 8.39.2.7 nChSizeX

```
uint8_t gslc_tsXTextbox::nChSizeX
```

Width of characters (pixels)

#### 8.39.2.8 nChSizeY

```
uint8_t gslc_tsXTextbox::nChSizeY
```

Height of characters (pixels)

#### 8.39.2.9 nCurPosX

```
uint8_t gslc_tsXTextbox::nCurPosX
```

Cursor X position.



**8.39.2.10 nCurPosY**

```
uint8_t gslc_tsXTextbox::nCurPosY
```

Cursor Y position.

**8.39.2.11 nMarginX**

```
int8_t gslc_tsXTextbox::nMarginX
```

Margin for text area within element rect (X)

**8.39.2.12 nMarginY**

```
int8_t gslc_tsXTextbox::nMarginY
```

Margin for text area within element rect (Y)

**8.39.2.13 nRedrawRow**

```
int16_t gslc_tsXTextbox::nRedrawRow
```

Specific row to update in redraw (if not -1)

**8.39.2.14 nScrollPos**

```
uint16_t gslc_tsXTextbox::nScrollPos
```

Current scrollbar position.

**8.39.2.15 nWndCols**

```
uint8_t gslc_tsXTextbox::nWndCols
```

Window X size.

#### 8.39.2.16 nWndRows

```
uint8_t gslc_tsXTextbox::nWndRows
```

Window Y size.

#### 8.39.2.17 nWndRowStart

```
uint8_t gslc_tsXTextbox::nWndRowStart
```

First row of current window.

#### 8.39.2.18 pBuf

```
char* gslc_tsXTextbox::pBuf
```

Ptr to the text buffer (circular buffer)

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

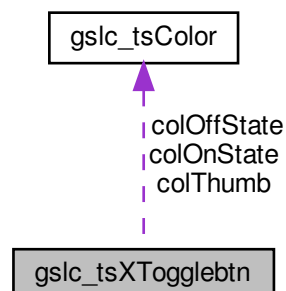
- [src/elem/XTextbox.h](#)

## 8.40 gslc\_tsXTogglebtn Struct Reference

Extended data for Togglebtn element.

```
#include <XTogglebtn.h>
```

Collaboration diagram for gslc\_tsXTogglebtn:



## Data Fields

- bool [bOn](#)  
*Indicates if button is ON or OFF.*
- int16\_t [nMyPageId](#)  
*We need to track our page in case of grouping elements on a non current layer, like base layer.*
- [gslc\\_tsColor](#) [colThumb](#)  
*Color of thumb.*
- [gslc\\_tsColor](#) [colOnState](#)  
*Color of button in ON state.*
- [gslc\\_tsColor](#) [colOffState](#)  
*Color of button in OFF state.*
- bool [bCircular](#)  
*Style of the toggle button circular or rectangular.*
- [GSLC\\_CB\\_TOUCH](#) [pfuncUser](#)  
*User's Callback event to say element has changed.*

### 8.40.1 Detailed Description

Extended data for Togglebtn element.

### 8.40.2 Field Documentation

#### 8.40.2.1 bCircular

```
bool gslc_tsXTogglebtn::bCircular
```

Style of the toggle button circular or rectangular.

#### 8.40.2.2 bOn

```
bool gslc_tsXTogglebtn::bOn
```

Indicates if button is ON or OFF.

#### 8.40.2.3 colOffState

```
gslc\_tsColor gslc_tsXTogglebtn::colOffState
```

Color of button in OFF state.

#### 8.40.2.4 colOnState

`gslc_tsColor` `gslc_tsXTogglebtn::colOnState`

Color of button in ON state.

#### 8.40.2.5 colThumb

`gslc_tsColor` `gslc_tsXTogglebtn::colThumb`

Color of thumb.

#### 8.40.2.6 nMyPageId

`int16_t` `gslc_tsXTogglebtn::nMyPageId`

We need to track our page in case of grouping elements on a non current layer, like base layer.

#### 8.40.2.7 pfunctUser

`GSLC_CB_TOUCH` `gslc_tsXTogglebtn::pfunctUser`

User's Callback event to say element has changed.

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

- [src/elem/XTogglebtn.h](#)

## 8.41 gslc\_tsXToggleImgbtn Struct Reference

Extended data for ToggleImgbtn element.

```
#include <XToggleImgbtn.h>
```

### Data Fields

- bool `bOn`  
*Indicates if button is ON or OFF.*
- `int16_t` `nMyPageId`  
*We need to track our page in case of grouping elements on a non current layer, like base layer.*
- `GSLC_CB_TOUCH` `pfunctUser`  
*User's Callback event to say element has changed.*

### 8.41.1 Detailed Description

Extended data for ToggleImgbtn element.

### 8.41.2 Field Documentation

#### 8.41.2.1 bOn

```
bool gslc_tsXToggleImgbtn::bOn
```

Indicates if button is ON or OFF.

#### 8.41.2.2 nMyPageId

```
int16_t gslc_tsXToggleImgbtn::nMyPageId
```

We need to track our page in case of grouping elements on a non current layer, like base layer.

#### 8.41.2.3 pfunctUser

```
GSLC_CB_TOUCH gslc_tsXToggleImgbtn::pfunctUser
```

User's Callback event to say element has changed.

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

- [src/elem/XToggleImgbtn.h](#)

## 8.42 THPoint Class Reference

```
#include <GUIslice_th.h>
```

### Public Member Functions

- [THPoint](#) (void)
- [THPoint](#) (uint16\_t x, uint16\_t y, uint16\_t z)
- bool [operator==](#) ([THPoint](#))
- bool [operator!=](#) ([THPoint](#))

## Data Fields

- uint16\_t [x](#)
- uint16\_t [y](#)
- uint16\_t [z](#)

## 8.42.1 Constructor & Destructor Documentation

### 8.42.1.1 THPoint() [1/2]

```
THPoint::THPoint (  
    void )
```

### 8.42.1.2 THPoint() [2/2]

```
THPoint::THPoint (  
    uint16_t x,  
    uint16_t y,  
    uint16_t z )
```

## 8.42.2 Member Function Documentation

### 8.42.2.1 operator!=(())

```
bool THPoint::operator!= (  
    THPoint p1 )
```

### 8.42.2.2 operator==(())

```
bool THPoint::operator== (  
    THPoint p1 )
```

## 8.42.3 Field Documentation

## 8.42.3.1 x

```
uint16_t THPoint::x
```

## 8.42.3.2 y

```
uint16_t THPoint::y
```

## 8.42.3.3 z

```
uint16_t THPoint::z
```

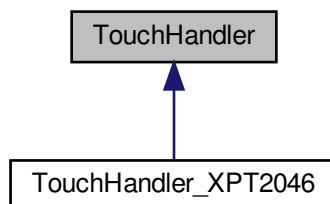
The documentation for this class was generated from the following files:

- [src/GUISlice\\_th.h](#)
- [src/GUISlice\\_th.cpp](#)

## 8.43 TouchHandler Class Reference

```
#include <GUISlice_th.h>
```

Inheritance diagram for TouchHandler:



### Public Member Functions

- [TouchHandler](#) ()
- void [setSize](#) (uint16\_t \_disp\_xSize, uint16\_t \_disp\_ySize)
- void [setCalibration](#) (uint16\_t ts\_xMin, uint16\_t ts\_xMax, uint16\_t ts\_yMin, uint16\_t ts\_yMax)
- void [setSwapFlip](#) (bool \_swapXY, bool \_flipX, bool \_flipY)
- [THPoint](#) [scale](#) ([THPoint](#) pIn)
- virtual void [begin](#) (void)
- virtual [THPoint](#) [getPoint](#) (void)

## 8.43.1 Constructor & Destructor Documentation

### 8.43.1.1 TouchHandler()

```
TouchHandler::TouchHandler ( ) [inline]
```

## 8.43.2 Member Function Documentation

### 8.43.2.1 begin()

```
void TouchHandler::begin (
    void ) [virtual]
```

Reimplemented in [TouchHandler\\_XPT2046](#).

### 8.43.2.2 getPoint()

```
THPoint TouchHandler::getPoint (
    void ) [virtual]
```

Reimplemented in [TouchHandler\\_XPT2046](#).

### 8.43.2.3 scale()

```
THPoint TouchHandler::scale (
    THPoint pIn )
```

### 8.43.2.4 setCalibration()

```
void TouchHandler::setCalibration (
    uint16_t ts_xMin,
    uint16_t ts_xMax,
    uint16_t ts_yMin,
    uint16_t ts_yMax )
```



#### 8.43.2.5 setSize()

```
void TouchHandler::setSize (
    uint16_t _disp_xSize,
    uint16_t _disp_ySize )
```

#### 8.43.2.6 setSwapFlip()

```
void TouchHandler::setSwapFlip (
    bool _swapXY,
    bool _flipX,
    bool _flipY )
```

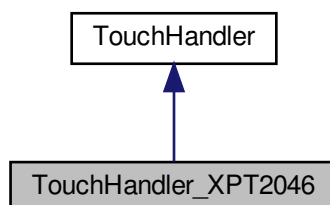
The documentation for this class was generated from the following files:

- [src/GUIslice\\_th.h](#)
- [src/GUIslice\\_th.cpp](#)

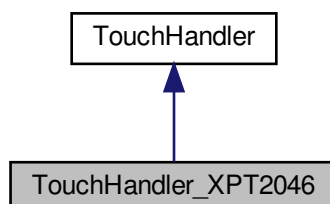
## 8.44 TouchHandler\_XPT2046 Class Reference

```
#include <GUIslice_th_XPT2046.h>
```

Inheritance diagram for TouchHandler\_XPT2046:



Collaboration diagram for TouchHandler\_XPT2046:



## Public Member Functions

- [TouchHandler\\_XPT2046](#) (SPIClass &[spi](#), uint8\_t [spi\\_cs\\_pin](#))
- void [begin](#) (void)
- [THPoint](#) [getPoint](#) (void)

## Data Fields

- SPIClass [spi](#)
- XPT2046\_touch [touchDriver](#)

## 8.44.1 Constructor & Destructor Documentation

### 8.44.1.1 TouchHandler\_XPT2046()

```
TouchHandler_XPT2046::TouchHandler_XPT2046 (
    SPIClass & spi,
    uint8_t spi_cs_pin ) [inline]
```

## 8.44.2 Member Function Documentation

### 8.44.2.1 begin()

```
void TouchHandler_XPT2046::begin (
    void ) [inline], [virtual]
```

Reimplemented from [TouchHandler](#).

### 8.44.2.2 getPoint()

```
THPoint TouchHandler_XPT2046::getPoint (
    void ) [inline], [virtual]
```

Reimplemented from [TouchHandler](#).

## 8.44.3 Field Documentation

#### 8.44.3.1 spi

```
SPIClass TouchHandler_XPT2046::spi
```

#### 8.44.3.2 touchDriver

```
XPT2046_touch TouchHandler_XPT2046::touchDriver
```

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

- [src/GUIslice\\_th\\_XPT2046.h](#)



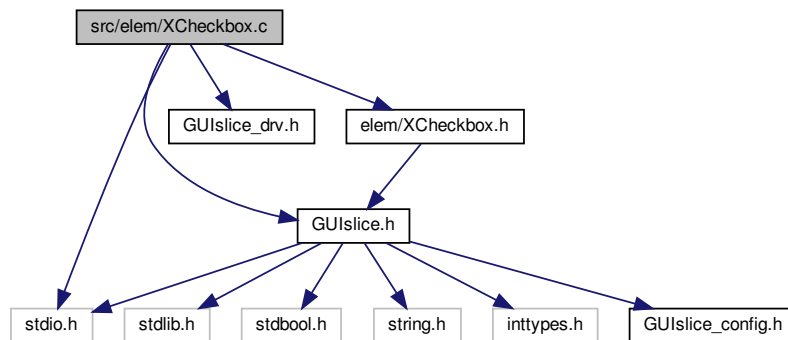
## Chapter 9

# File Documentation

### 9.1 README.md File Reference

### 9.2 src/elem/XCheckbox.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XCheckbox.h"
#include <stdio.h>
Include dependency graph for XCheckbox.c:
```



### Functions

- `gslc_tsElemRef * gslc_ElemXCheckboxCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXCheckbox *pXData, gslc\_tsRect rElem, bool bRadio, gslc\_teXCheckboxStyle nStyle, gslc\_tsColor col, bool bChecked)`  
*Create a Checkbox or Radio button Element.*
- `bool gslc\_ElemXCheckboxGetState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
*Get a Checkbox element's current state.*
- `gslc\_tsElemRef * gslc\_ElemXCheckboxFindChecked (gslc_tsGui *pGui, int16_t nGroupId)`

*Find the checkbox within a group that has been checked.*

- void [gslc\\_ElemXCheckboxSetStateFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_XCHECKBOX](#) pfuncCb)

*Assign the state callback function for a checkbox/radio button.*

- void [gslc\\_ElemXCheckboxSetStateHelp](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bChecked, bool bDoCb)
- void [gslc\\_ElemXCheckboxSetState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bChecked)

*Set a Checkbox element's current state.*

- void [gslc\\_ElemXCheckboxToggleState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Toggle a Checkbox element's current state.*

- bool [gslc\\_ElemXCheckboxDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw a Checkbox element on the screen.*

- bool [gslc\\_ElemXCheckboxTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)

*Handle touch events to Checkbox element.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.2.1 Function Documentation

### 9.2.1.1 [gslc\\_ElemXCheckboxCreate\(\)](#)

```
gslc\_tsElemRef* gslc\_ElemXCheckboxCreate (
    gslc\_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc\_tsXCheckbox * pXData,
    gslc\_tsRect rElem,
    bool bRadio,
    gslc\_teXCheckboxStyle nStyle,
    gslc\_tsColor colCheck,
    bool bChecked )
```

Create a Checkbox or Radio button Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or <a href="#">GSLC_ID_AUTO</a> to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>bRadio</i>	Radio-button functionality if true
in	<i>nStyle</i>	Drawing style for checkbox / radio button
in	<i>colCheck</i>	Color for inner fill when checked
in	<i>bChecked</i>	Default state

**Returns**

Pointer to Element reference or NULL if failure

**9.2.1.2 gslc\_ElemXCheckboxDraw()**

```
bool gslc_ElemXCheckboxDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Checkbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.2.1.3 gslc\_ElemXCheckboxFindChecked()**

```
gslc_tsElemRef* gslc_ElemXCheckboxFindChecked (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```

Find the checkbox within a group that has been checked.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ GroupId</i>	Group ID to search

**Returns**

Element Ptr or NULL if none checked

#### 9.2.1.4 gslc\_ElemXCheckboxGetState()

```
bool gslc_ElemXCheckboxGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Checkbox element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

##### Returns

Current state

#### 9.2.1.5 gslc\_ElemXCheckboxSetState()

```
void gslc_ElemXCheckboxSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bChecked )
```

Set a Checkbox element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bChecked</i>	New state

##### Returns

none

#### 9.2.1.6 gslc\_ElemXCheckboxSetStateFunc()

```
void gslc_ElemXCheckboxSetStateFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XCHECKBOX pfuncCb )
```

Assign the state callback function for a checkbox/radio button.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>pfuncCb</i>	Function pointer to callback routine (or NULL for none)

## Returns

none

## 9.2.1.7 gslc\_ElemXCheckboxSetStateHelp()

```
void gslc_ElemXCheckboxSetStateHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bChecked,
    bool bDoCb )
```

## 9.2.1.8 gslc\_ElemXCheckboxToggleState()

```
void gslc_ElemXCheckboxToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a Checkbox element's current state.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

none

## 9.2.1.9 gslc\_ElemXCheckboxTouch()

```
bool gslc_ElemXCheckboxTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Checkbox element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

## 9.2.2 Variable Documentation

### 9.2.2.1 ERRSTR\_NULL

```
const char ERRSTR_NULL
```

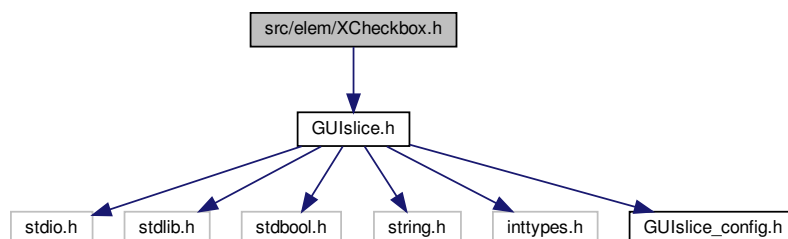
### 9.2.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```

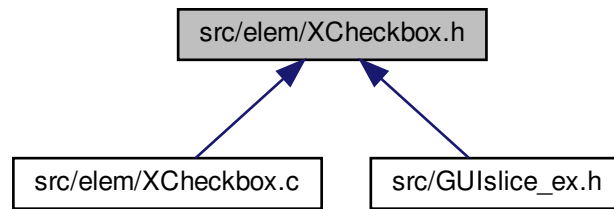
## 9.3 src/elem/XCheckbox.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XCheckbox.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXCheckbox](#)  
*Extended data for Checkbox element.*

## Macros

- #define [GSLC\\_TYPEX\\_CHECKBOX](#)
- #define [gslc\\_ElemXCheckboxCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, colFill, bFillEn, nGroup, bRadio\_, nStyle\_, colCheck\_, bChecked\_)  
*Create a Checkbox or Radio button Element in Flash.*

## Typedefs

- typedef bool(\* [GSLC\\_CB\\_XCHECKBOX](#)) (void \*pvGui, void \*pvElemRef, int16\_t nSelId, bool bChecked)  
*Callback function for checkbox/radio element state change.*

## Enumerations

- enum [gslc\\_teXCheckboxStyle](#) { [GSLCX\\_CHECKBOX\\_STYLE\\_BOX](#), [GSLCX\\_CHECKBOX\\_STYLE\\_X](#), [GSLCX\\_CHECKBOX\\_STYLE\\_ROUND](#) }  
*Checkbox drawing style.*

## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXCheckboxCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXCheckbox](#) \*pXData, [gslc\\_tsRect](#) rElem, bool bRadio, [gslc\\_teXCheckboxStyle](#) nStyle, [gslc\\_tsColor](#) colCheck, bool bChecked)  
*Create a Checkbox or Radio button Element.*
- bool [gslc\\_ElemXCheckboxGetState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a Checkbox element's current state.*
- void [gslc\\_ElemXCheckboxSetState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bChecked)  
*Set a Checkbox element's current state.*

- `gslc_tsElemRef * gslc_ElemXCheckboxFindChecked (gslc_tsGui *pGui, int16_t nGroupId)`  
*Find the checkbox within a group that has been checked.*
- `void gslc_ElemXCheckboxToggleState (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`  
*Toggle a Checkbox element's current state.*
- `void gslc_ElemXCheckboxSetStateFunc (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XCHECKBOX pfuncCb)`  
*Assign the state callback function for a checkbox/radio button.*
- `bool gslc_ElemXCheckboxDraw (void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`  
*Draw a Checkbox element on the screen.*
- `bool gslc_ElemXCheckboxTouch (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
*Handle touch events to Checkbox element.*

### 9.3.1 Macro Definition Documentation

#### 9.3.1.1 `gslc_ElemXCheckboxCreate_P`

```
#define gslc_ElemXCheckboxCreate_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    colFill,  
    bFillEn,  
    nGroup,  
    bRadio_,  
    nStyle_,  
    colCheck_,  
    bChecked_ )
```

Create a Checkbox or Radio button Element in Flash.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>colFill</i>	Color for the control background fill
in	<i>bFillEn</i>	True if background filled, false otherwise (recommend True)
in	<i>nGroup</i>	Group ID that radio buttons belong to (else GSLC_GROUP_NONE)
in	<i>bRadio_</i>	Radio-button functionality if true
in	<i>nStyle_</i>	Drawing style for checkbox / radio button

## Parameters

in	<i>col</i> ↔ <i>Check_</i>	Color for inner fill when checked
in	<i>b</i> ↔ <i>Checked</i> ↔ —	Default state

## Returns

none

## 9.3.1.2 GSLC\_TYPEX\_CHECKBOX

```
#define GSLC_TYPEX_CHECKBOX
```

## 9.3.2 Typedef Documentation

## 9.3.2.1 GSLC\_CB\_XCHECKBOX

```
typedef bool(* GSLC_CB_XCHECKBOX) (void *pvGui, void *pvElemRef, int16_t nSelId, bool bChecked)
```

Callback function for checkbox/radio element state change.

- nSelId: Selected element's ID or GSLC\_ID\_NONE
- bChecked: Element was selected if true, false otherwise

## 9.3.3 Enumeration Type Documentation

## 9.3.3.1 gslc\_teXCheckboxStyle

```
enum gslc_teXCheckboxStyle
```

Checkbox drawing style.

## Enumerator

GSLCX_CHECKBOX_STYLE_BOX	Inner box.
GSLCX_CHECKBOX_STYLE_X	Crossed.
GSLCX_CHECKBOX_STYLE_ROUND	Circular.

### 9.3.4 Function Documentation

#### 9.3.4.1 `gslc_ElemXCheckboxCreate()`

```
gslc_tsElemRef* gslc_ElemXCheckboxCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXCheckbox * pXData,
    gslc_tsRect rElem,
    bool bRadio,
    gslc_teXCheckboxStyle nStyle,
    gslc_tsColor colCheck,
    bool bChecked )
```

Create a Checkbox or Radio button Element.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>bRadio</i>	Radio-button functionality if true
in	<i>nStyle</i>	Drawing style for checkbox / radio button
in	<i>colCheck</i>	Color for inner fill when checked
in	<i>bChecked</i>	Default state

##### Returns

Pointer to Element reference or NULL if failure

#### 9.3.4.2 `gslc_ElemXCheckboxDraw()`

```
bool gslc_ElemXCheckboxDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Checkbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

## 9.3.4.3 gslc\_ElemXCheckboxFindChecked()

```
gslc_tsElemRef* gslc_ElemXCheckboxFindChecked (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```

Find the checkbox within a group that has been checked.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ GroupId</i>	Group ID to search

## Returns

Element Ptr or NULL if none checked

## 9.3.4.4 gslc\_ElemXCheckboxGetState()

```
bool gslc_ElemXCheckboxGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Checkbox element's current state.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

Current state

#### 9.3.4.5 gslc\_ElemXCheckboxSetState()

```
void gslc_ElemXCheckboxSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bChecked )
```

Set a Checkbox element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bChecked</i>	New state

##### Returns

none

#### 9.3.4.6 gslc\_ElemXCheckboxSetStateFunc()

```
void gslc_ElemXCheckboxSetStateFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XCHECKBOX pfuncCb )
```

Assign the state callback function for a checkbox/radio button.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>pfuncCb</i>	Function pointer to callback routine (or NULL for none)

##### Returns

none

#### 9.3.4.7 gslc\_ElemXCheckboxToggleState()

```
void gslc_ElemXCheckboxToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a Checkbox element's current state.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

none

## 9.3.4.8 gslc\_ElemXCheckboxTouch()

```
bool gslc_ElemXCheckboxTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Checkbox element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

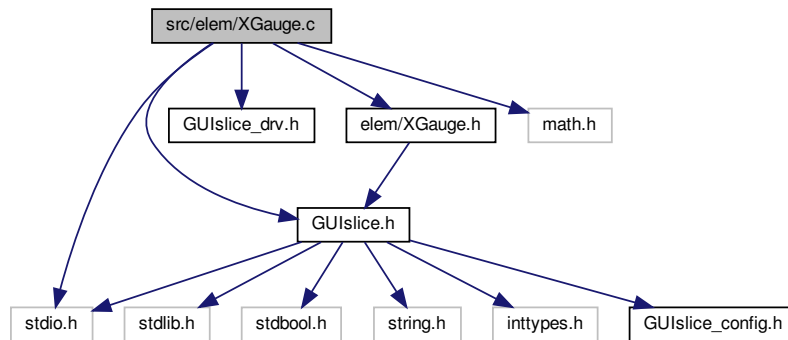
## Returns

true if success, false otherwise

## 9.4 src/elem/XGauge.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XGauge.h"
#include <stdio.h>
#include <math.h>
```

Include dependency graph for XGauge.c:



## Functions

- `gslc_tsElemRef * gslc_ElemXGaugeCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXGauge *pXData, gslc\_tsRect rElem, int16_t nMin, int16_t nMax, int16_t nVal, gslc\_tsColor colGauge, bool bVert)`  
Create a Gauge Element.
- `void gslc\_ElemXGaugeSetStyle (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_teXGaugeStyle nStyle)`  
Configure the style of a Gauge element.
- `void gslc\_ElemXGaugeSetIndicator (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colGauge, uint16_t nIndicLen, uint16_t nIndicTip, bool bIndicFill)`  
Configure the appearance of the Gauge indicator.
- `void gslc\_ElemXGaugeSetTicks (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colTick, uint16_t nTickCnt, uint16_t nTickLen)`  
Configure the appearance of the Gauge ticks.
- `void gslc\_ElemXGaugeUpdate (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nVal)`  
Update a Gauge element's current value.
- `void gslc\_ElemXGaugeSetFlip (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bFlip)`  
Set a Gauge element's fill direction.
- `bool gslc\_ElemXGaugeDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
Draw a gauge element on the screen.
- `bool gslc\_ElemXGaugeDrawProgressBar (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_teRedrawType eRedraw)`  
Helper function to draw a gauge with style: progress bar.

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

### 9.4.1 Function Documentation

9.4.1.1 `gslc_ElemXGaugeCreate()`

```
gslc_tsElemRef* gslc_ElemXGaugeCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGauge * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )
```

Create a Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.
- Support gauge sub-types:
  - `GSLC_TYPEX_GAUGE_PROG_BAR`: Horizontal or vertical box with filled region
  - `GSLC_TYPEX_GAUGE_RADIAL`: Radial / compass indicator
- Default appearance is a horizontal progress bar, but can be changed with `gslc_ElemXGaugeSetStyle()`

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or <code>GSLC_ID_AUTO</code> to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

## Returns

Pointer to Element reference or NULL if failure

9.4.1.2 `gslc_ElemXGaugeDraw()`

```
bool gslc_ElemXGaugeDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from `gslc_ElemDraw()`

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.4.1.3 `gslc_ElemXGaugeDrawProgressBar()`**

```
bool gslc_ElemXGaugeDrawProgressBar (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: progress bar.

- Called from [gslc\\_ElemXGaugeDraw\(\)](#)

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

**Returns**

true if success, false otherwise

**9.4.1.4 `gslc_ElemXGaugeSetFlip()`**

```
void gslc_ElemXGaugeSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )
```

Set a Gauge element's fill direction.

- Setting `bFlip` reverses the default fill direction
- Default fill direction for horizontal gauges: left-to-right
- Default fill direction for vertical gauges: bottom-to-top

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of fill from default

## Returns

none

## 9.4.1.5 gslc\_ElemXGaugeSetIndicator()

```
void gslc_ElemXGaugeSetIndicator (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge,
    uint16_t nIndicLen,
    uint16_t nIndicTip,
    bool bIndicFill )
```

Configure the appearance of the Gauge indicator.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colGauge</i>	Color of the indicator
in	<i>nIndicLen</i>	Length of the indicator
in	<i>nIndicTip</i>	Size of the indicator tip
in	<i>bIndicFill</i>	Fill in the indicator if true

## Returns

none

## 9.4.1.6 gslc\_ElemXGaugeSetStyle()

```
void gslc_ElemXGaugeSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teXGaugeStyle nType )
```

Configure the style of a Gauge element.

- This function is used to select between one of several gauge types (eg. progress bar, radial dial, etc.)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nType</i>	Gauge style enumeration

**Returns**

none

**9.4.1.7 gslc\_ElemXGaugeSetTicks()**

```
void gslc_ElemXGaugeSetTicks (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colTick,
    uint16_t nTickCnt,
    uint16_t nTickLen )
```

Configure the appearance of the Gauge ticks.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colTick</i>	Color of the gauge ticks
in	<i>nTickCnt</i>	Number of ticks to draw around / along gauge
in	<i>nTickLen</i>	Length of the tick marks to draw

**Returns**

none

**9.4.1.8 gslc\_ElemXGaugeUpdate()**

```
void gslc_ElemXGaugeUpdate (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

## Returns

none

## 9.4.2 Variable Documentation

## 9.4.2.1 ERRSTR\_NULL

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

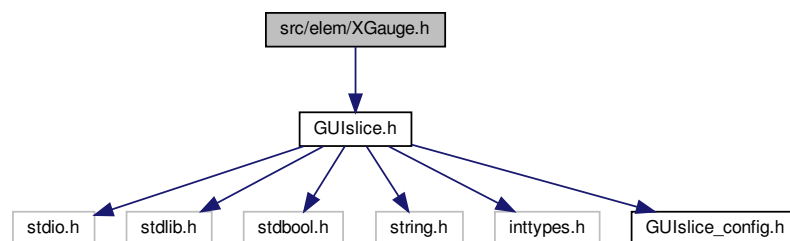
## 9.4.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```

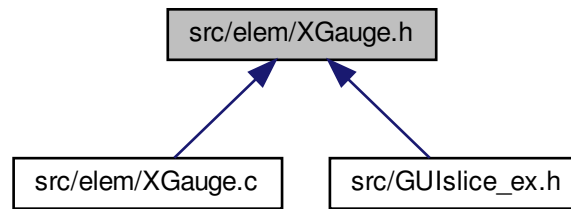
## 9.5 src/elem/XGauge.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XGauge.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXGauge](#)  
*Extended data for Gauge element.*

## Macros

- #define [GSLC\\_TYPEX\\_GAUGE](#)
- #define [gslc\\_ElemXGaugeCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, nMin\_, nMax\_, nVal\_, colFrame\_, colFill\_, colGauge\_, bVert\_)  
*Create a Gauge Element in Flash.*

## Enumerations

- enum [gslc\\_teXGaugeStyle](#) { [GSLCX\\_GAUGE\\_STYLE\\_PROG\\_BAR](#), [GSLCX\\_GAUGE\\_STYLE\\_RADIAL](#), [GSLCX\\_GAUGE\\_STYLE\\_RAMP](#) }  
*Gauge drawing style.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXGaugeCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXGauge](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nMin, int16\_t nMax, int16\_t nVal, [gslc\\_tsColor](#) colGauge, bool bVert)  
*Create a Gauge Element.*
- void [gslc\\_ElemXGaugeSetStyle](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teXGaugeStyle](#) nType)  
*Configure the style of a Gauge element.*
- void [gslc\\_ElemXGaugeSetIndicator](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colGauge, uint16\_t nIndicLen, uint16\_t nIndicTip, bool bIndicFill)  
*Configure the appearance of the Gauge indicator.*
- void [gslc\\_ElemXGaugeSetTicks](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colTick, uint16\_t nTickCnt, uint16\_t nTickLen)  
*Configure the appearance of the Gauge ticks.*
- void [gslc\\_ElemXGaugeUpdate](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Update a Gauge element's current value.*



- void [gslc\\_ElemXGaugeSetFlip](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFlip)  
*Set a Gauge element's fill direction.*
- bool [gslc\\_ElemXGaugeDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a gauge element on the screen.*
- bool [gslc\\_ElemXGaugeDrawProgressBar](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Helper function to draw a gauge with style: progress bar.*

## 9.5.1 Macro Definition Documentation

### 9.5.1.1 [gslc\\_ElemXGaugeCreate\\_P](#)

```
#define gslc_ElemXGaugeCreate_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    nMin_,  
    nMax_,  
    nVal_,  
    colFrame_,  
    colFill_,  
    colGauge_,  
    bVert_ )
```

Create a Gauge Element in Flash.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nMin_</i>	Minimum value of gauge for nVal comparison
in	<i>nMax_</i>	Maximum value of gauge for nVal comparison
in	<i>nVal_</i>	Starting value of gauge
in	<i>colFrame_</i>	Color for the gauge frame
in	<i>colFill_</i>	Color for the gauge background fill
in	<i>colGauge_</i>	Color for the gauge indicator
in	<i>bVert_</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

**Returns**

none

**9.5.1.2 GSLC\_TYPEX\_GAUGE**

```
#define GSLC_TYPEX_GAUGE
```

**9.5.2 Enumeration Type Documentation****9.5.2.1 gslc\_tXGaugeStyle**

```
enum gslc_tXGaugeStyle
```

Gauge drawing style.

**Enumerator**

GSLCX_GAUGE_STYLE_PROG_BAR	Progress bar.
GSLCX_GAUGE_STYLE_RADIAL	Radial indicator.
GSLCX_GAUGE_STYLE_RAMP	Ramp indicator.

**9.5.3 Function Documentation****9.5.3.1 gslc\_ElemXGaugeCreate()**

```
gslc_tsElemRef* gslc_ElemXGaugeCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGauge * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )
```

Create a Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.
- Support gauge sub-types:
  - GSLC\_TYPEX\_GAUGE\_PROG\_BAR: Horizontal or vertical box with filled region
  - GSLC\_TYPEX\_GAUGE\_RADIAL: Radial / compass indicator
- Default appearance is a horizontal progress bar, but can be changed with [gslc\\_ElemXGaugeSetStyle\(\)](#)

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

**Returns**

Pointer to Element reference or NULL if failure

**9.5.3.2 gslc\_ElemXGaugeDraw()**

```
bool gslc_ElemXGaugeDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

### 9.5.3.3 gslc\_ElemXGaugeDrawProgressBar()

```
bool gslc_ElemXGaugeDrawProgressBar (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: progress bar.

- Called from [gslc\\_ElemXGaugeDraw\(\)](#)

#### Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

#### Returns

true if success, false otherwise

### 9.5.3.4 gslc\_ElemXGaugeSetFlip()

```
void gslc_ElemXGaugeSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )
```

Set a Gauge element's fill direction.

- Setting bFlip reverses the default fill direction
- Default fill direction for horizontal gauges: left-to-right
- Default fill direction for vertical gauges: bottom-to-top

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of fill from default

#### Returns

none

## 9.5.3.5 gslc\_ElemXGaugeSetIndicator()

```
void gslc_ElemXGaugeSetIndicator (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge,
    uint16_t nIndicLen,
    uint16_t nIndicTip,
    bool bIndicFill )
```

Configure the appearance of the Gauge indicator.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colGauge</i>	Color of the indicator
in	<i>nIndicLen</i>	Length of the indicator
in	<i>nIndicTip</i>	Size of the indicator tip
in	<i>bIndicFill</i>	Fill in the indicator if true

## Returns

none

## 9.5.3.6 gslc\_ElemXGaugeSetStyle()

```
void gslc_ElemXGaugeSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teXGaugeStyle nType )
```

Configure the style of a Gauge element.

- This function is used to select between one of several gauge types (eg. progress bar, radial dial, etc.)

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nType</i>	Gauge style enumeration

## Returns

none

### 9.5.3.7 gslc\_ElemXGaugeSetTicks()

```
void gslc_ElemXGaugeSetTicks (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colTick,
    uint16_t nTickCnt,
    uint16_t nTickLen )
```

Configure the appearance of the Gauge ticks.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colTick</i>	Color of the gauge ticks
in	<i>nTickCnt</i>	Number of ticks to draw around / along gauge
in	<i>nTickLen</i>	Length of the tick marks to draw

#### Returns

none

### 9.5.3.8 gslc\_ElemXGaugeUpdate()

```
void gslc_ElemXGaugeUpdate (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

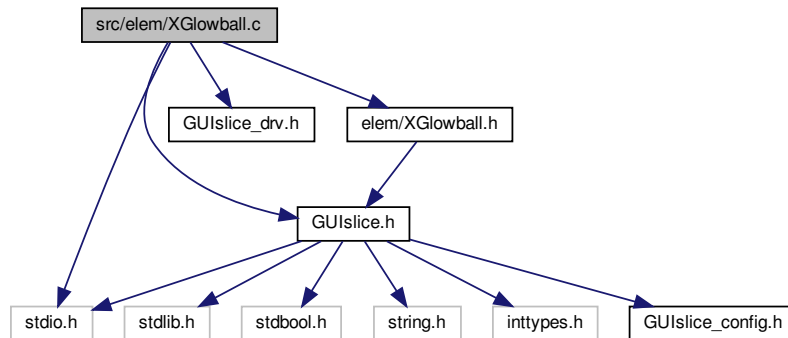
#### Returns

none

## 9.6 src/elem/XGlowball.c File Reference

```
#include "GUIslice.h"
```

```
#include "GUIslice_drv.h"
#include "elem/XGlowball.h"
#include <stdio.h>
Include dependency graph for XGlowball.c:
```



## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXGlowballCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXGlowball](#) \*pXData, int16\_t nMidX, int16\_t nMidY, [gslc\\_tsXGlowballRing](#) \*pRings, uint8\_t nNumRings)

*Create a XGlowball element.*

- void [drawXGlowballArc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArc, uint16\_t nAngStart, uint16\_t nAngEnd)
- void [drawXGlowballRing](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nVal, uint16\_t nAngStart, uint16\_t nAngEnd, bool bErase)
- void [drawXGlowball](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nVal, uint16\_t nAngStart, uint16\_t nAngEnd)
- void [gslc\\_ElemXGlowballSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)
- void [gslc\\_ElemXGlowballSetAngles](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nAngStart, int16\_t nAngEnd)
- void [gslc\\_ElemXGlowballSetQuality](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nQuality)
- void [gslc\\_ElemXGlowballSetColorBack](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colBg)
- bool [gslc\\_ElemXGlowballDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw the XGlowball element on the screen.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

### 9.6.1 Function Documentation

#### 9.6.1.1 drawXGlowball()

```
void drawXGlowball (
    gslc_tsGui * pGui,
    gslc_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nVal,
    uint16_t nAngStart,
    uint16_t nAngEnd )
```

#### 9.6.1.2 drawXGlowballArc()

```
void drawXGlowballArc (
    gslc_tsGui * pGui,
    gslc_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nRad1,
    int16_t nRad2,
    gslc_tsColor cArc,
    uint16_t nAngStart,
    uint16_t nAngEnd )
```

#### 9.6.1.3 drawXGlowballRing()

```
void drawXGlowballRing (
    gslc_tsGui * pGui,
    gslc_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nVal,
    uint16_t nAngStart,
    uint16_t nAngEnd,
    bool bErase )
```

#### 9.6.1.4 gslc\_ElemXGlowballCreate()

```
gslc_tsElemRef* gslc_ElemXGlowballCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGlowball * pXData,
    int16_t nMidX,
    int16_t nMidY,
    gslc_tsXGlowballRing * pRings,
    uint8_t nNumRings )
```

Create a XGlowball element.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nMidX</i>	Center X coordinate
in	<i>nMidY</i>	Center Y coordinate
in	<i>pRings</i>	Pointer to tsXGlowballRing structure array defining appearance
in	<i>nNumRings</i>	Number of rings in pRings array

## Returns

Pointer to Element reference or NULL if failure

## 9.6.1.5 gslc\_ElemXGlowballDraw()

```
bool gslc_ElemXGlowballDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw the XGlowball element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

## 9.6.1.6 gslc\_ElemXGlowballSetAngles()

```
void gslc_ElemXGlowballSetAngles (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nAngStart,
    int16_t nAngEnd )
```

#### 9.6.1.7 gslc\_ElemXGlowballSetColorBack()

```
void gslc_ElemXGlowballSetColorBack (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colBg )
```

#### 9.6.1.8 gslc\_ElemXGlowballSetQuality()

```
void gslc_ElemXGlowballSetQuality (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nQuality )
```

#### 9.6.1.9 gslc\_ElemXGlowballSetVal()

```
void gslc_ElemXGlowballSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

### 9.6.2 Variable Documentation

#### 9.6.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

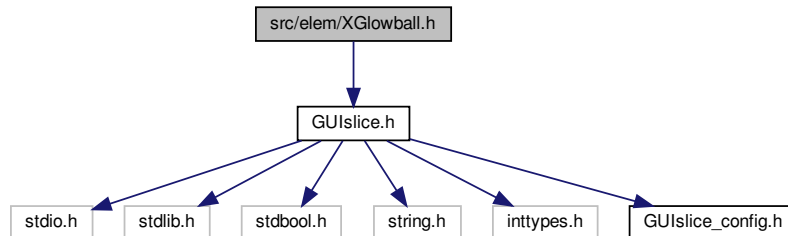
#### 9.6.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

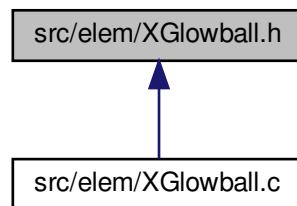
## 9.7 src/elem/XGlowball.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XGlowball.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gslc\\_tsXGlowballRing](#)
- struct [gslc\\_tsXGlowball](#)

*Extended data for Slider element.*

### Macros

- `#define` [GSLC\\_TYPEX\\_GLOW](#)

### Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXGlowballCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXGlowball](#) \*pXData, int16\_t nMidX, int16\_t nMidY, [gslc\\_tsXGlowballRing](#) \*pRings, uint8\_t nNumRings)
- *Create a XGlowball element.*
- bool [gslc\\_ElemXGlowballDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw the XGlowball element on the screen.*

- void [drawXGlowballArc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArc, uint16\_t nAngStart, uint16\_t nAngEnd)
- void [drawXGlowballRing](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nVal, uint16\_t nAngStart, uint16\_t nAngEnd, bool bErase)
- void [drawXGlowball](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXGlowball](#) \*pGlowball, int16\_t nMidX, int16\_t nMidY, int16\_t nVal, uint16\_t nAngStart, uint16\_t nAngEnd)
- void [gslc\\_ElemXGlowballSetAngles](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nAngStart, int16\_t nAngEnd)
- void [gslc\\_ElemXGlowballSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)
- void [gslc\\_ElemXGlowballSetQuality](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nQuality)
- void [gslc\\_ElemXGlowballSetColorBack](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colBg)

## 9.7.1 Macro Definition Documentation

### 9.7.1.1 GSLC\_TYPEX\_GLOW

```
#define GSLC_TYPEX_GLOW
```

## 9.7.2 Function Documentation

### 9.7.2.1 drawXGlowball()

```
void drawXGlowball (
    gslc\_tsGui * pGui,
    gslc\_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nVal,
    uint16_t nAngStart,
    uint16_t nAngEnd )
```

### 9.7.2.2 drawXGlowballArc()

```
void drawXGlowballArc (
    gslc\_tsGui * pGui,
    gslc\_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nRad1,
    int16_t nRad2,
    gslc\_tsColor cArc,
    uint16_t nAngStart,
    uint16_t nAngEnd )
```

## 9.7.2.3 drawXGlowballRing()

```
void drawXGlowballRing (
    gslc_tsGui * pGui,
    gslc_tsXGlowball * pGlowball,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nVal,
    uint16_t nAngStart,
    uint16_t nAngEnd,
    bool bErase )
```

## 9.7.2.4 gslc\_ElemXGlowballCreate()

```
gslc_tsElemRef* gslc_ElemXGlowballCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGlowball * pXData,
    int16_t nMidX,
    int16_t nMidY,
    gslc_tsXGlowballRing * pRings,
    uint8_t nNumRings )
```

Create a XGlowball element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nMidX</i>	Center X coordinate
in	<i>nMidY</i>	Center Y coordinate
in	<i>pRings</i>	Pointer to tsXGlowballRing structure array defining appearance
in	<i>nNumRings</i>	Number of rings in pRings array

## Returns

Pointer to Element reference or NULL if failure

## 9.7.2.5 gslc\_ElemXGlowballDraw()

```
bool gslc_ElemXGlowballDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw the XGlowball element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

#### 9.7.2.6 `gslc_ElemXGlowballSetAngles()`

```
void gslc_ElemXGlowballSetAngles (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nAngStart,
    int16_t nAngEnd )
```

#### 9.7.2.7 `gslc_ElemXGlowballSetColorBack()`

```
void gslc_ElemXGlowballSetColorBack (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colBg )
```

#### 9.7.2.8 `gslc_ElemXGlowballSetQuality()`

```
void gslc_ElemXGlowballSetQuality (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nQuality )
```

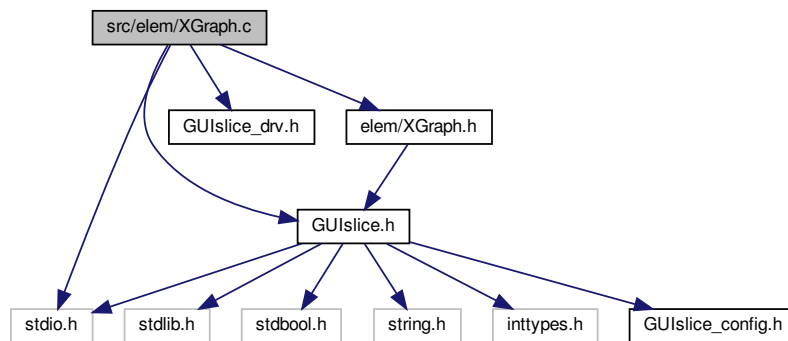
#### 9.7.2.9 `gslc_ElemXGlowballSetVal()`

```
void gslc_ElemXGlowballSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

## 9.8 src/elem/XGraph.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XGraph.h"
#include <stdio.h>
```

Include dependency graph for XGraph.c:



### Functions

- `gslc_tsElemRef * gslc_ElemXGraphCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXGraph *pXData, gslc\_tsRect rElem, int16_t nFontId, int16_t *pBuf, uint16_t nBufMax, gslc\_tsColor colGraph)`  
Create a Graph Element.
- `void gslc\_ElemXGraphSetStyle (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_teXGraphStyle eStyle, uint8_t nMargin)`  
Set the graph's additional drawing characteristics.
- `void gslc\_ElemXGraphSetRange (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nYMin, int16_t nYMax)`  
Set the graph's drawing range.
- `void gslc\_ElemXGraphScrollSet (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)`  
Set the graph scroll position (nScrollPos) as a fraction of nScrollMax.
- `void gslc\_ElemXGraphAdd (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nVal)`  
Add a value to the graph at the latest position.
- `void gslc\_ElemXGraphReset (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
Reset the graphs values.
- `bool gslc\_ElemXGraphDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
Draw a Graph element on the screen.

### Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

## 9.8.1 Function Documentation

### 9.8.1.1 gslc\_ElemXGraphAdd()

```
void gslc_ElemXGraphAdd (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Add a value to the graph at the latest position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	Data value to add

#### Returns

none

### 9.8.1.2 gslc\_ElemXGraphCreate()

```
gslc_tsElemRef* gslc_ElemXGraphCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGraph * pXData,
    gslc_tsRect rElem,
    int16_t nFontId,
    int16_t * pBuf,
    uint16_t nBufRows,
    gslc_tsColor colGraph )
```

Create a Graph Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nFontId</i>	Font ID to use for graph area
in	<i>pBuf</i>	Ptr to data buffer (already allocated) with size (nBufMax) int16_t
in	<i>nBufRows</i>	Maximum number of points in buffer
in	<i>colGraph</i>	Color of the graph



**Returns**

Pointer to Element reference or NULL if failure

**9.8.1.3 gslc\_ElemXGraphDraw()**

```
bool gslc_ElemXGraphDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Graph element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.8.1.4 gslc\_ElemXGraphReset()**

```
void gslc_ElemXGraphReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Reset the graphs values.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

none

### 9.8.1.5 gslc\_ElemXGraphScrollSet()

```
void gslc_ElemXGraphScrollSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nScrollPos,
    uint8_t nScrollMax )
```

Set the graph scroll position (nScrollPos) as a fraction of nScrollMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	New scroll position
in	<i>nScrollMax</i>	Maximum scroll position

#### Returns

none

### 9.8.1.6 gslc\_ElemXGraphSetRange()

```
void gslc_ElemXGraphSetRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nYMin,
    int16_t nYMax )
```

Set the graph's drawing range.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nYMin</i>	Minimum Y value to draw
in	<i>nYMax</i>	Maximum Y value to draw

#### Returns

none

### 9.8.1.7 gslc\_ElemXGraphSetStyle()

```
void gslc_ElemXGraphSetStyle (
    gslc_tsGui * pGui,
```

```

gslc_tsElemRef * pElemRef,
gslc_tXGraphStyle eStyle,
uint8_t nMargin )

```

Set the graph's additional drawing characteristics.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>eStyle</i>	Drawing style for the graph
in	<i>nMargin</i>	Margin to provide around graph area inside frame

#### Returns

none

## 9.8.2 Variable Documentation

### 9.8.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

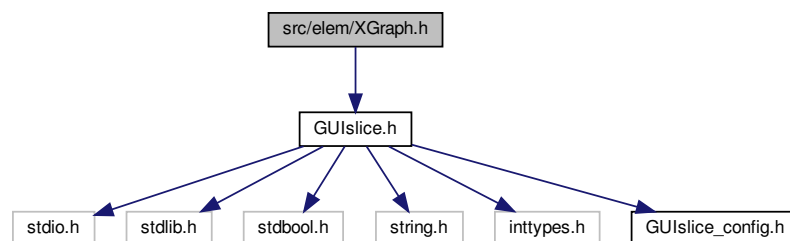
### 9.8.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

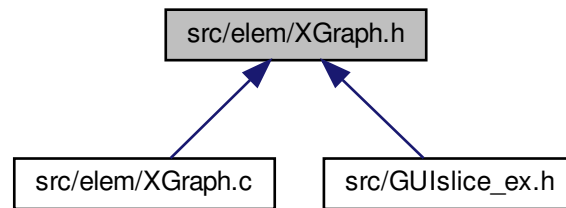
## 9.9 src/elem/XGraph.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XGraph.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXGraph](#)  
*Extended data for Graph element.*

## Macros

- #define [GSLC\\_TYPEX\\_GRAPH](#)

## Enumerations

- enum [gslc\\_teXGraphStyle](#) { [GSLCX\\_GRAPH\\_STYLE\\_DOT](#), [GSLCX\\_GRAPH\\_STYLE\\_LINE](#), [GSLCX\\_GRAPH\\_STYLE\\_FILL](#) }  
*Gauge drawing style.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXGraphCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXGraph](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nFontId, int16\_t \*pBuf, uint16\_t nBufRows, [gslc\\_tsColor](#) col, [gslc\\_tsXGraph](#) \*pGraph)  
*Create a Graph Element.*
- void [gslc\\_ElemXGraphSetStyle](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teXGraphStyle](#) eStyle, uint8\_t nMargin)  
*Set the graph's additional drawing characteristics.*
- void [gslc\\_ElemXGraphSetRange](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nYMin, int16\_t nYMax)  
*Set the graph's drawing range.*
- bool [gslc\\_ElemXGraphDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Graph element on the screen.*
- void [gslc\\_ElemXGraphAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Add a value to the graph at the latest position.*
- void [gslc\\_ElemXGraphReset](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Reset the graphs values.*
- void [gslc\\_ElemXGraphScrollSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nScrollPos, uint8\_t nScrollMax)  
*Set the graph scroll position (nScrollPos) as a fraction of nScrollMax.*

## 9.9.1 Macro Definition Documentation

### 9.9.1.1 GSLC\_TYPEX\_GRAPH

```
#define GSLC_TYPEX_GRAPH
```

## 9.9.2 Enumeration Type Documentation

### 9.9.2.1 gslc\_tXGraphStyle

```
enum gslc_tXGraphStyle
```

Gauge drawing style.

#### Enumerator

GSLCX_GRAPH_STYLE_DOT	Dot.
GSLCX_GRAPH_STYLE_LINE	Line.
GSLCX_GRAPH_STYLE_FILL	Filled.

## 9.9.3 Function Documentation

### 9.9.3.1 gslc\_ElemXGraphAdd()

```
void gslc_ElemXGraphAdd (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Add a value to the graph at the latest position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	Data value to add

**Returns**

none

**9.9.3.2 gslc\_ElemXGraphCreate()**

```
gslc_tsElemRef* gslc_ElemXGraphCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXGraph * pXData,
    gslc_tsRect rElem,
    int16_t nFontId,
    int16_t * pBuf,
    uint16_t nBufRows,
    gslc_tsColor colGraph )
```

Create a Graph Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nFontId</i>	Font ID to use for graph area
in	<i>pBuf</i>	Ptr to data buffer (already allocated) with size (nBufMax) int16_t
in	<i>nBufRows</i>	Maximum number of points in buffer
in	<i>colGraph</i>	Color of the graph

**Returns**

Pointer to Element reference or NULL if failure

**9.9.3.3 gslc\_ElemXGraphDraw()**

```
bool gslc_ElemXGraphDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Graph element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

## 9.9.3.4 gslc\_ElemXGraphReset()

```
void gslc_ElemXGraphReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Reset the graphs values.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

none

## 9.9.3.5 gslc\_ElemXGraphScrollSet()

```
void gslc_ElemXGraphScrollSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nScrollPos,
    uint8_t nScrollMax )
```

Set the graph scroll position (nScrollPos) as a fraction of nScrollMax.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	New scroll position
in	<i>nScrollMax</i>	Maximum scroll position

**Returns**

none

**9.9.3.6 gslc\_ElemXGraphSetRange()**

```
void gslc_ElemXGraphSetRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nYMin,
    int16_t nYMax )
```

Set the graph's drawing range.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nYMin</i>	Minimum Y value to draw
in	<i>nYMax</i>	Maximum Y value to draw

**Returns**

none

**9.9.3.7 gslc\_ElemXGraphSetStyle()**

```
void gslc_ElemXGraphSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_texGraphStyle eStyle,
    uint8_t nMargin )
```

Set the graph's additional drawing characteristics.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>eStyle</i>	Drawing style for the graph
in	<i>nMargin</i>	Margin to provide around graph area inside frame

**Returns**

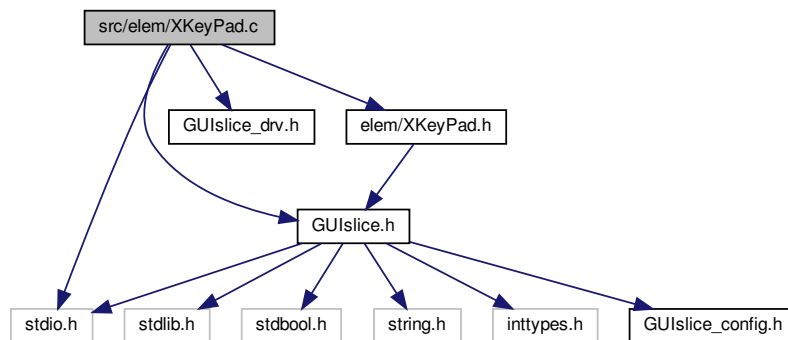
none



## 9.10 src/elem/XKeyPad.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XKeyPad.h"
#include <stdio.h>
```

Include dependency graph for XKeyPad.c:



### Functions

- void [gslc\\_ElemXKeyPadReset](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad)
- void [gslc\\_ElemXKeyPadCfgInit](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig)
 

*Provide default initialization for the base XKeyPad.*
- int16\_t [gslc\\_XKeyPadLookupId](#) ([gslc\\_tsKey](#) \*pKeys, uint8\_t nKeyId)
 

*Find a key ID within a KeyPad label array and return it's index into the array.*
- int16\_t [gslc\\_XKeyPadLookupSpecialId](#) ([gslc\\_tsLabelSpecial](#) \*pLabels, uint8\_t nKeyId)
 

*Find a key ID within a KeyPad special label array and return it's index into the array.*
- void [gslc\\_XKeyPadDrawLayout](#) ([gslc\\_tsGui](#) \*pGui, void \*pXData)
- void [gslc\\_XKeyPadDrawKey](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXKeyPad](#) \*pXData, [gslc\\_tsKey](#) \*pKey, bool bGlow, bool bFocus)
 

*Draw a key to the screen.*
- [gslc\\_tsElemRef](#) \* [gslc\\_XKeyPadCreateBase](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXKeyPad](#) \*pXData, int16\_t nX0, int16\_t nY0, int8\_t nFontId, [gslc\\_tsXKeyPadCfg](#) \*pConfig)
 

*Create a KeyPad Element.*
- void [gslc\\_XKeyPadAdjustScroll](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad)
- bool [gslc\\_XKeyPadLayoutSet](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig, int8\_t eLayoutSel)
 

*Select a new KeyPad layout.*
- void [gslc\\_ElemXKeyPadValSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStrBuf)
 

*Set the current value for the editable text field.*
- void [gslc\\_ElemXKeyPadTargetRefSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsElemRef](#) \*p←TxtRef)
 

*Set target element reference for KeyPad return value.*
- int16\_t [gslc\\_ElemXKeyPadDataTargetIdGet](#) ([gslc\\_tsGui](#) \*pGui, void \*pvData)
 

*Fetch the element target ID associated with this KeyPad.*
- char \* [gslc\\_ElemXKeyPadDataValGet](#) ([gslc\\_tsGui](#) \*pGui, void \*pvData)
 

*Fetch the final value string of the KeyPad from a callback.*

- bool [gslc\\_ElemXKeyPadValGet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, char \*pStrBuf, uint8\_t nStrBufLen)
 

*Fetch the current value string associated with KeyPad element.*
- bool [gslc\\_XKeyPadDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)
 

*Draw a KeyPad element on the screen.*
- void [gslc\\_ElemXKeyPadValSetCb](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_INPUT](#) pfuncCb)
 

*Set the callback function associated with the KeyPad.*
- void [gslc\\_XKeyPadSizeAllGet](#) ([gslc\\_tsKey](#) \*\*pLayouts, uint8\_t nNumLayouts, uint8\_t \*pnRows, uint8\_t \*pnCols)
 

*Calculate the overall dimensions of the KeyPad control encompassing all available layouts for the KeyPad, leveraging the computation in [gslc\\_XKeyPadSizeGet](#)().*
- void [gslc\\_XKeyPadSizeGet](#) ([gslc\\_tsKey](#) \*pLayout, uint8\_t \*pnRows, uint8\_t \*pnCols, int8\_t \*pnIndFirst, int8\_t \*pnIndLast)
 

*Calculate the overall dimensions of the KeyPad control encompassing the text field and key buttons.*
- int16\_t [gslc\\_XKeyPadMapEvent](#) ([gslc\\_tsGui](#) \*pGui, void \*pXData, int16\_t nRelX, int16\_t nRelY, int16\_t \*pnInd)
- void [gslc\\_XKeyPadPendRedrawReset](#) ([gslc\\_tsXKeyPadResult](#) \*pResult)
- void [gslc\\_XKeyPadPendRedrawAddTxt](#) ([gslc\\_tsXKeyPadResult](#) \*pResult)
- void [gslc\\_XKeyPadPendRedrawAddKey](#) ([gslc\\_tsXKeyPadResult](#) \*pResult, int16\_t nId)
- void [gslc\\_XKeyPadRedrawUpdate](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- void [gslc\\_XKeyPadTrackSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nInd, [gslc\\_tsXKeyPadAttrib](#) eAttrib)
- void [gslc\\_XKeyPadFocusSetDefault](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- bool [gslc\\_XKeyPadTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nX, int16\_t nY)
 

*Handle touch (up,down,move) events to KeyPad element.*
- bool [gslc\\_XKeyPadTxtDelCh](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, uint8\_t nPos)
 

*Remove a character from the KeyPad text field at the specified offset (nPos).*
- bool [gslc\\_XKeyPadTxtAddCh](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, char ch, uint8\_t nPos)
 

*Add a character to the KeyPad text field at the specified offset (nPos).*
- bool [gslc\\_XKeyPadTxtAddStr](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, const char \*pStr, uint8\_t nPos)
 

*Add a string to the KeyPad text field at the specified offset (nPos).*
- void [gslc\\_ElemXKeyPadCfgSetButtonSz](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig, int8\_t nButtonSzW, int8\_t nButtonSzH)
 

*Update the KeyPad configuration to define the KeyPad button sizing.*
- void [gslc\\_ElemXKeyPadCfgSetButtonSpace](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig, int8\_t nSpaceX, int8\_t nSpaceY)
 

*Update the KeyPad configuration to define the KeyPad button spacing.*
- void [gslc\\_ElemXKeyPadCfgSetRoundEn](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig, bool bEn)
 

*Update the KeyPad configuration to enable rounded button corners.*
- void [gslc\\_XKeyPadDrawVirtualTxt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rElem, [gslc\\_tsXKeyPad](#) \*pKeyPad, [gslc\\_tsColor](#) cColFrame, [gslc\\_tsColor](#) cColFill, [gslc\\_tsColor](#) cColTxt)
 

*Draw a virtual Text Element.*
- void [gslc\\_XKeyPadDrawVirtualBtn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId, [gslc\\_tsColor](#) cColFrame, [gslc\\_tsColor](#) cColFill, [gslc\\_tsColor](#) cColFillGlow, [gslc\\_tsColor](#) cColText, bool bRoundedEn, bool bGlow, bool bFocus)
 

*Draw a virtual textual Button Element.*
- void [gslc\\_ElemXKeyPadInputAsk](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pKeyPadRef, int16\_t nPgPopup, [gslc\\_tsElemRef](#) \*pTxtRef)
 

*Trigger a KeyPad popup and associate it with a text element.*
- char \* [gslc\\_ElemXKeyPadInputGet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pTxtRef, void \*pvCbData)
 

*Complete a KeyPad popup by retrieving the input data and storing it in the text element.*

## Variables

- const char [GSLC\\_PMEM ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM ERRSTR\\_PXD\\_NULL](#) []

## 9.10.1 Function Documentation

### 9.10.1.1 gslc\_ElemXKeyPadCfgInit()

```
void gslc_ElemXKeyPadCfgInit (
    gslc\_tsXKeyPadCfg * pConfig )
```

Provide default initialization for the base XKeyPad.

- These defaults will be overwritten by variant-specific initialization or through user configuration APIs.

#### Parameters

in	<i>pConfig</i>	Ptr to the KeyPad base config structure
----	----------------	---

#### Returns

none

### 9.10.1.2 gslc\_ElemXKeyPadCfgSetButtonSpace()

```
void gslc_ElemXKeyPadCfgSetButtonSpace (
    gslc\_tsXKeyPadCfg * pConfig,
    int8_t nSpaceX,
    int8_t nSpaceY )
```

Update the KeyPad configuration to define the KeyPad button spacing.

- This defines the inset amount (X and Y) from the Button Size
- A spacing of (1,1) will mean that the button is drawn with a 1 pixel margin around the grid defined by the Button Size

#### Parameters

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>nSpaceX</i>	Amount to inset button in horizontal direction (pixels)
in	<i>nSpaceY</i>	Amount to inset button in vertical direction (pixels)

**Returns**

none

**9.10.1.3 gslc\_ElemXKeyPadCfgSetButtonSz()**

```
void gslc_ElemXKeyPadCfgSetButtonSz (
    gslc_tsXKeyPadCfg * pConfig,
    int8_t nButtonSzW,
    int8_t nButtonSzH )
```

Update the KeyPad configuration to define the KeyPad button sizing.

**Parameters**

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>nButtonSzW</i>	Width of buttons in pixels
in	<i>nButtonSzH</i>	Height of buttons in pixels

**Returns**

none

**9.10.1.4 gslc\_ElemXKeyPadCfgSetRoundEn()**

```
void gslc_ElemXKeyPadCfgSetRoundEn (
    gslc_tsXKeyPadCfg * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable rounded button corners.

**Parameters**

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>bEn</i>	Enable rounded corners

**Returns**

none

**9.10.1.5 gslc\_ElemXKeyPadDataTargetIdGet()**

```
int16_t gslc_ElemXKeyPadDataTargetIdGet (
    gslc_tsGui * pGui,
    void * pvData )
```

Fetch the element target ID associated with this KeyPad.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pvData</i>	: Void ptr to callback data structure

#### Returns

Target Element ID or GSLC\_ID\_NONE if unspecified

#### 9.10.1.6 gslc\_ElemXKeyPadDataValGet()

```
char* gslc_ElemXKeyPadDataValGet (
    gslc_tsGui * pGui,
    void * pvData )
```

Fetch the final value string of the KeyPad from a callback.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
out	<i>pvData</i>	: Void ptr to callback data structure

#### Returns

Pointer to edited character string

#### 9.10.1.7 gslc\_ElemXKeyPadInputAsk()

```
void gslc_ElemXKeyPadInputAsk (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pKeyPadRef,
    int16_t nPgPopup,
    gslc_tsElemRef * pTxtRef )
```

Trigger a KeyPad popup and associate it with a text element.

- This function also updates the maximum KeyPad buffer length to match that of the target text element, up to the maximum XKEYPAD\_BUF\_MAX.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pKeyPadRef</i>	Pointer to KeyPad element reference
in	<i>nPgPopup</i>	Page enum that contains the popup to show
in	<i>pTxtRef</i>	Pointer to associated text field element reference

**Returns**

none

**9.10.1.8 gslc\_ElemXKeyPadInputGet()**

```
char* gslc_ElemXKeyPadInputGet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pTxtRef,
    void * pvCbData )
```

Complete a KeyPad popup by retrieving the input data and storing it in the text element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pTxtRef</i>	Pointer to associated text field element reference
in	<i>pvCbData</i>	Void pointer to callback function's pvData

**Returns**

The text string that was fetched from the KeyPad (NULL terminated)

**9.10.1.9 gslc\_ElemXKeyPadReset()**

```
void gslc_ElemXKeyPadReset (
    gslc_tsXKeyPad * pKeyPad )
```

**9.10.1.10 gslc\_ElemXKeyPadTargetRefSet()**

```
void gslc_ElemXKeyPadTargetRefSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsElemRef * pTargetRef )
```

Set target element reference for KeyPad return value.

- The Target Reference is used in the GSLC\_CB\_INPUT callback so that the user has the context needed to determine which field should be edited with the contents of the KeyPad edit field
- It is expected that the user will call this function when showing the KeyPad popup dialog

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
in	<i>pTargetRef</i>	Element reference for target of KeyPad value

## Returns

none

## 9.10.1.11 gslc\_ElemXKeyPadValGet()

```
bool gslc_ElemXKeyPadValGet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    char * pStrBuf,
    uint8_t nStrBufMax )
```

Fetch the current value string associated with KeyPad element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
out	<i>pStrBuf</i>	String buffer
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf) including terminator

## Returns

true if success, false otherwise

## 9.10.1.12 gslc\_ElemXKeyPadValSet()

```
void gslc_ElemXKeyPadValSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    const char * pStrBuf )
```

Set the current value for the editable text field.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
in	<i>pStrBuf</i>	String to copy into keypad

**Returns**

none

**9.10.1.13 gslc\_ElemXKeyPadValSetCb()**

```
void gslc_ElemXKeyPadValSetCb (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_INPUT pfuncCb )
```

Set the callback function associated with the KeyPad.

- This function will be called during updates and OK / Cancel

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element Reference for KeyPad
in	<i>pfuncCb</i>	Callback function pointer

**Returns**

none

**9.10.1.14 gslc\_XKeyPadAdjustScroll()**

```
void gslc_XKeyPadAdjustScroll (
    gslc_tsXKeyPad * pKeyPad )
```

**9.10.1.15 gslc\_XKeyPadCreateBase()**

```
gslc_tsElemRef* gslc_XKeyPadCreateBase (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pXData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg * pConfig )
```

Create a KeyPad Element.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Pointer to base Config options

## Returns

Pointer to Element or NULL if failure

## 9.10.1.16 gslc\_XKeyPadDraw()

```
bool gslc_XKeyPadDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a KeyPad element on the screen.

- Called during redraw

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

## 9.10.1.17 gslc\_XKeyPadDrawKey()

```
void gslc_XKeyPadDrawKey (
    gslc_tsGui * pGui,
    gslc_tsXKeyPad * pXData,
    gslc_tsKey * pKey,
```

```
bool bGlow,  
bool bFocus )
```

Draw a key to the screen.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>pKey</i>	Ptr to key being drawn
in	<i>bGlow</i>	Indicate if key is in glow state
in	<i>bFocus</i>	Indicate if key is in focus state

## Returns

none

## 9.10.1.18 gslc\_XKeyPadDrawLayout()

```
void gslc_XKeyPadDrawLayout (
    gslc_tsGui * pGui,
    void * pXData )
```

## 9.10.1.19 gslc\_XKeyPadDrawVirtualBtn()

```
void gslc_XKeyPadDrawVirtualBtn (
    gslc_tsGui * pGui,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId,
    gslc_tsColor cColFrame,
    gslc_tsColor cColFill,
    gslc_tsColor cColFillGlow,
    gslc_tsColor cColTxt,
    bool bRoundedEn,
    bool bGlow,
    bool bFocus )
```

Draw a virtual textual Button Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pStrBuf</i>	String to copy into element
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf).
in	<i>nFontId</i>	Font ID to use for text display
in	<i>cColFrame</i>	Frame color for element
in	<i>cColFill</i>	Fill color for element
in	<i>cColFillGlow</i>	Fill color for element when glowing/focused
in	<i>cColTxt</i>	Text color for element
in	<i>bRoundedEn</i>	Use Rounded Corners?
in	<i>bGlow</i>	Indicate btn is in glow state
in	<i>bFocus</i>	Indicate btn is in focus state

**Returns**

none

**9.10.1.20 gslc\_XKeyPadDrawVirtualTxt()**

```
void gslc_XKeyPadDrawVirtualTxt (
    gslc_tsGui * pGui,
    gslc_tsRect rElem,
    gslc_tsXKeyPad * pKeyPad,
    gslc_tsColor cColFrame,
    gslc_tsColor cColFill,
    gslc_tsColor cColTxt )
```

Draw a virtual Text Element.

- Creates a text string with filled background

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pKeyPad</i>	Pointer to KeyPad struct
in	<i>cColFrame</i>	Frame color for element
in	<i>cColFill</i>	Fill color for element
in	<i>cColTxt</i>	Text color for element

**Returns**

none

**9.10.1.21 gslc\_XKeyPadFocusSetDefault()**

```
void gslc_XKeyPadFocusSetDefault (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

**Todo** Doc

## 9.10.1.22 gslc\_XKeyPadLayoutSet()

```
bool gslc_XKeyPadLayoutSet (
    gslc_tsXKeyPadCfg * pConfig,
    int8_t eLayoutSel )
```

Select a new KeyPad layout.

- Multiple KeyPad layouts can share the same key key definition array (eg. KEYPAD\_LAYOUT)
- This function returns an indication of whether a full KeyPad control redraw is required, ie. the KeyPad layout definition has changed. With changes in the KeyPad definition, there may be a different number of visible keys or arrangements, necessitating a background redraw.

## Parameters

in	<i>pConfig</i>	Ptr to the KeyPad configuration
in	<i>eLayoutSel</i>	Layout index to select

## Returns

true if a full redraw should be performed

## 9.10.1.23 gslc\_XKeyPadLookupId()

```
int16_t gslc_XKeyPadLookupId (
    gslc_tsKey * pKeys,
    uint8_t nKeyId )
```

Find a key ID within a KeyPad label array and return it's index into the array.

- It is expected that the KeyPad label array is terminated with KEYPAD\_ID\_\_END

## Parameters

in	<i>pKeys</i>	Ptr to the Keypad label array
in	<i>n↔ KeyId</i>	Key ID to look for

## Returns

the index into the array if the ID was found or -1 if the key ID was not found

#### 9.10.1.24 gslc\_XKeyPadLookupSpecialId()

```
int16_t gslc_XKeyPadLookupSpecialId (
    gslc_tsLabelSpecial * pLabels,
    uint8_t nKeyId )
```

Find a key ID within a KeyPad special label array and return it's index into the array.

- It is expected that the KeyPad label array is terminated with KEYPAD\_ID\_\_END

##### Parameters

in	<i>pLabels</i>	Ptr to the Keypad special label array
in	<i>nKeyId</i>	Key ID to look for

##### Returns

the index into the array if the ID was found or -1 if the key ID was not found

#### 9.10.1.25 gslc\_XKeyPadMapEvent()

```
int16_t gslc_XKeyPadMapEvent (
    gslc_tsGui * pGui,
    void * pXData,
    int16_t nRelX,
    int16_t nRelY,
    int16_t * pnInd )
```

#### 9.10.1.26 gslc\_XKeyPadPendRedrawAddKey()

```
void gslc_XKeyPadPendRedrawAddKey (
    gslc_tsXKeyPadResult * pResult,
    int16_t nId )
```

**Todo** Doc

#### 9.10.1.27 gslc\_XKeyPadPendRedrawAddTxt()

```
void gslc_XKeyPadPendRedrawAddTxt (
    gslc_tsXKeyPadResult * pResult )
```

**Todo** Doc

9.10.1.28 `gslc_XKeyPadPendRedrawReset()`

```
void gslc_XKeyPadPendRedrawReset (
    gslc\_tsXKeyPadResult * pResult )
```

**Todo** Doc

9.10.1.29 `gslc_XKeyPadRedrawUpdate()`

```
void gslc_XKeyPadRedrawUpdate (
    gslc\_tsGui * pGui,
    gslc\_tsElemRef * pElemRef )
```

**Todo** Doc

9.10.1.30 `gslc_XKeyPadSizeAllGet()`

```
void gslc_XKeyPadSizeAllGet (
    gslc\_tsKey ** pLayouts,
    uint8_t nNumLayouts,
    uint8_t * pnRows,
    uint8_t * pnCols )
```

Calculate the overall dimensions of the KeyPad control encompassing all available layouts for the KeyPad, leveraging the computation in [gslc\\_XKeyPadSizeGet\(\)](#).

## Parameters

in	<i>pLayouts</i>	Ptr to the array of KeyPad layouts
in	<i>nNumLayouts</i>	Number of layouts in pLayouts
out	<i>pnRows</i>	Ptr for the number of rows
out	<i>pnCols</i>	Ptr for the number of columns

## Returns

none

9.10.1.31 `gslc_XKeyPadSizeGet()`

```
void gslc_XKeyPadSizeGet (
    gslc\_tsKey * pLayout,
```

```

uint8_t * pnRows,
uint8_t * pnCols,
int8_t * pnIndFirst,
int8_t * pnIndLast )

```

Calculate the overall dimensions of the KeyPad control encompassing the text field and key buttons.

The dimension is calculated in units of the configured key size (width and height), and accounts for any column spans. It also returns the index of the first and last keys on the keypad.

#### Parameters

in	<i>pLayout</i>	Ptr to the KeyPad layout
out	<i>pnRows</i>	Ptr for the number of rows
out	<i>pnCols</i>	Ptr for the number of columns
out	<i>pnIndFirst</i>	Ptr for the index of first key
out	<i>pnIndLast</i>	Ptr for the index of last key

#### Returns

none

#### 9.10.1.32 gslc\_XKeyPadTouch()

```

bool gslc_XKeyPadTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )

```

Handle touch (up,down,move) events to KeyPad element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise



## 9.10.1.33 gslc\_XKeyPadTrackSet()

```
void gslc_XKeyPadTrackSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nInd,
    gslc_tsXKeyPadAttrib eAttrib )
```

**Todo** Doc

## 9.10.1.34 gslc\_XKeyPadTxtAddCh()

```
bool gslc_XKeyPadTxtAddCh (
    gslc_tsXKeyPad * pKeyPad,
    char ch,
    uint8_t nPos )
```

Add a character to the KeyPad text field at the specified offset (nPos).

Providing an offset equal to the end of the existing buffer length will cause the addition to the end, whereas an offset within the buffer will cause an insert.

- An addition that causes the buffer length to exceed the maximum allowed will result in the end of the resulting buffer to be truncated.
- Typically the addition will be done at the current cursor position.
- If the insertion is ahead of the cursor, then the cursor position may be increased.

## Parameters

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>ch</i>	Character to add
in	<i>nPos</i>	Buffer position for the insertion

## Returns

true if the text field should be redrawn, false if no redraw is needed (ie. no change)

## 9.10.1.35 gslc\_XKeyPadTxtAddStr()

```
bool gslc_XKeyPadTxtAddStr (
    gslc_tsXKeyPad * pKeyPad,
    const char * pStr,
    uint8_t nPos )
```

Add a string to the KeyPad text field at the specified offset (*nPos*).

Providing an offset equal to the end of the existing buffer length will cause the addition to the end, whereas an offset within the buffer will cause an insert.

- An addition that causes the buffer length to exceed the maximum allowed will result in the end of the resulting buffer to be truncated.
- Typically the addition will be done at the current cursor position.
- If the insertion is ahead of the cursor, then the cursor position may be increased.
- This routine may be useful when adding multi-byte characters for future support of foreign characters.

#### Parameters

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>pStr</i>	String to add
in	<i>nPos</i>	Buffer position for the insertion

#### Returns

true if the text field should be redrawn, false if no redraw is needed (ie. no change)

#### 9.10.1.36 gslc\_XKeyPadTxtDelCh()

```
bool gslc_XKeyPadTxtDelCh (
    gslc_tsXKeyPad * pKeyPad,
    uint8_t nPos )
```

Remove a character from the KeyPad text field at the specified offset (*nPos*).

- Typically the addition will be done at the current cursor position.
- If the removal is ahead of the cursor, then the cursor position may be decreased.

#### Parameters

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>nPos</i>	Buffer position for the removal

#### Returns

true if the text field should be redrawn, false if no redraw is needed (ie. no change)

### 9.10.2 Variable Documentation

## 9.10.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

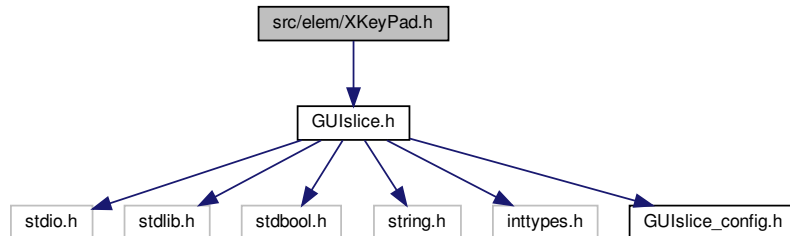
## 9.10.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

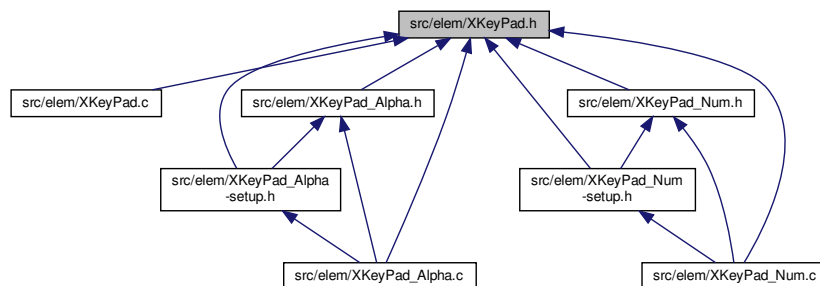
## 9.11 src/elem/XKeyPad.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XKeyPad.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXKeyPadResult](#)  
*Return status for XKeyPad.*
- struct [gslc\\_tsKey](#)  
*Key information. Defines everything we need to know about a particular key.*
- struct [gslc\\_tsLabelSpecial](#)  
*Key Label strings for special buttons.*
- struct [gslc\\_tsXKeyPadCfg](#)  
*Configuration for the KeyPad.*
- struct [gslc\\_tsXKeyPadData](#)  
*Input callback data structure.*
- struct [gslc\\_tsXKeyPad](#)  
*Extended data for KeyPad element.*

## Macros

- `#define XKEYPAD_BUF_MAX`
- `#define XKEYPAD_KEY_LEN`
- `#define XKEYPAD_CURSOR_ENHANCED`
- `#define XKEYPAD_CURSOR_CH`
- `#define GSLC_TYPEX_KEYPAD`
- `#define XKEYPAD_CB_STATE_DONE`
- `#define XKEYPAD_CB_STATE_CANCEL`
- `#define XKEYPAD_CB_STATE_UPDATE`
- `#define XKEYPAD_REDRAW_NONE`
- `#define XKEYPAD_REDRAW_TXT`
- `#define XKEYPAD_REDRAW_KEY`
- `#define XKEYPAD_REDRAW_ALL`
- `#define XKEYPAD_REDRAW_FULL`
- `#define DEBUG_XKEYPAD`  
*Debug message for XKeyPad (1=enabled, 0=disabled)*

## Typedefs

- typedef struct [gslc\\_tsKey](#) [gslc\\_tsKey](#)  
*Key information. Defines everything we need to know about a particular key.*
- typedef void(\* [GSLC\\_CB\\_XKEYPAD\\_RESET](#)) (void \*pvKeyPadConfig)
- typedef void(\* [GSLC\\_CB\\_XKEYPAD\\_TXT\\_INIT](#)) (void \*pvKeyPad)
- typedef void(\* [GSLC\\_CB\\_XKEYPAD\\_LABEL\\_GET](#)) (void \*pvKeyPad, uint8\_t nId, uint8\_t nStrMax, char \*pStr)
- typedef void(\* [GSLC\\_CB\\_XKEYPAD\\_SYTLE\\_GET](#)) (void \*pvKeyPad, uint8\_t nId, bool \*bVisible, [gslc\\_tsColor](#) \*pcolTxt, [gslc\\_tsColor](#) \*pcolFrame, [gslc\\_tsColor](#) \*pcolFill, [gslc\\_tsColor](#) \*pcolGlow)
- typedef void(\* [GSLC\\_CB\\_XKEYPAD\\_BTN\\_EVT](#)) (void \*pvKeyPad, uint8\_t nId, [gslc\\_tsXKeyPadResult](#) \*pResult)

## Enumerations

- enum {  
[KEYPAD\\_ID\\_BACKSPACE](#), [KEYPAD\\_ID\\_SPACE](#), [KEYPAD\\_ID\\_ESC](#), [KEYPAD\\_ID\\_ENTER](#),  
[KEYPAD\\_ID\\_SWAP\\_PAD](#), [KEYPAD\\_ID\\_SCROLL\\_LEFT](#), [KEYPAD\\_ID\\_SCROLL\\_RIGHT](#), [KEYPAD\\_ID\\_BASIC\\_START](#),  
[KEYPAD\\_ID\\_TXT](#), [KEYPAD\\_ID\\_\\_END](#) }
- enum {  
[E\\_XKEYPAD\\_TYPE\\_BASIC](#), [E\\_XKEYPAD\\_TYPE\\_SPECIAL](#), [E\\_XKEYPAD\\_TYPE\\_TXT](#), [E\\_XKEYPAD\\_TYPE\\_UNUSED](#),  
[E\\_XKEYPAD\\_TYPE\\_END](#) }
- enum [gslc\\_tsXKeyPadAttrib](#) { [E\\_XKEYPAD\\_ATTRIB\\_FOCUS](#), [E\\_XKEYPAD\\_ATTRIB\\_GLOW](#) }

## Functions

- void [gslc\\_ElemXKeyPadCfgInit](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig)  
*Provide default initialization for the base XKeyPad.*
- int16\_t [gslc\\_XKeyPadLookupId](#) ([gslc\\_tsKey](#) \*pKeys, uint8\_t nKeyId)  
*Find a key ID within a KeyPad label array and return it's index into the array.*
- int16\_t [gslc\\_XKeyPadLookupSpecialId](#) ([gslc\\_tsLabelSpecial](#) \*pLabels, uint8\_t nKeyId)  
*Find a key ID within a KeyPad special label array and return it's index into the array.*
- bool [gslc\\_XKeyPadTxtAddCh](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, char ch, uint8\_t nPos)  
*Add a character to the KeyPad text field at the specified offset (nPos).*
- bool [gslc\\_XKeyPadTxtAddStr](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, const char \*pStr, uint8\_t nPos)  
*Add a string to the KeyPad text field at the specified offset (nPos).*
- bool [gslc\\_XKeyPadTxtDelCh](#) ([gslc\\_tsXKeyPad](#) \*pKeyPad, uint8\_t nPos)  
*Remove a character from the KeyPad text field at the specified offset (nPos).*
- bool [gslc\\_XKeyPadLayoutSet](#) ([gslc\\_tsXKeyPadCfg](#) \*pConfig, int8\_t eLayoutSel)  
*Select a new KeyPad layout.*
- void [gslc\\_XKeyPadSizeAllGet](#) ([gslc\\_tsKey](#) \*\*pLayouts, uint8\_t nNumLayouts, uint8\_t \*pnRows, uint8\_t \*pnCols)  
*Calculate the overall dimensions of the KeyPad control encompassing all available layouts for the KeyPad, leveraging the computation in [gslc\\_XKeyPadSizeGet\(\)](#).*
- void [gslc\\_XKeyPadSizeGet](#) ([gslc\\_tsKey](#) \*pLayout, uint8\_t \*pnRows, uint8\_t \*pnCols, int8\_t \*pnIndFirst, int8\_t \*pnIndLast)  
*Calculate the overall dimensions of the KeyPad control encompassing the text field and key buttons.*
- void [gslc\\_XKeyPadDrawKey](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXKeyPad](#) \*pXData, [gslc\\_tsKey](#) \*pKey, bool bGlow, bool bFocus)  
*Draw a key to the screen.*
- [gslc\\_tsElemRef](#) \* [gslc\\_XKeyPadCreateBase](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXKeyPad](#) \*pXData, int16\_t nX0, int16\_t nY0, int8\_t nFontId, [gslc\\_tsXKeyPadCfg](#) \*pConfig)  
*Create a KeyPad Element.*
- void [gslc\\_ElemXKeyPadValSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStrBuf)  
*Set the current value for the editable text field.*
- void [gslc\\_ElemXKeyPadTargetRefSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsElemRef](#) \*pTargetRef)  
*Set target element reference for KeyPad return value.*
- bool [gslc\\_ElemXKeyPadValGet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, char \*pStrBuf, uint8\_t nStrBufMax)  
*Fetch the current value string associated with KeyPad element.*
- char \* [gslc\\_ElemXKeyPadDataValGet](#) ([gslc\\_tsGui](#) \*pGui, void \*pvData)  
*Fetch the final value string of the KeyPad from a callback.*

- `int16_t gslc_ElemXKeyPadDataTargetIdGet (gslc_tsGui *pGui, void *pvData)`  
*Fetch the element target ID associated with this KeyPad.*
- `void gslc_XKeyPadPendRedrawReset (gslc_tsXKeyPadResult *pResult)`
- `void gslc_XKeyPadPendRedrawAddKey (gslc_tsXKeyPadResult *pResult, int16_t nId)`
- `void gslc_XKeyPadPendRedrawAddTxt (gslc_tsXKeyPadResult *pResult)`
- `void gslc_XKeyPadTrackSet (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nInd, gslc_tsXKeyPad↵  
Attrib eAttrib)`
- `void gslc_XKeyPadFocusSetDefault (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`
- `void gslc_XKeyPadRedrawUpdate (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`
- `bool gslc_XKeyPadDraw (void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`  
*Draw a KeyPad element on the screen.*
- `bool gslc_XKeyPadTouch (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t n↵  
RelY)`  
*Handle touch (up,down,move) events to KeyPad element.*
- `void gslc_ElemXKeyPadValSetCb (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_INPUT  
pfuncCb)`  
*Set the callback function associated with the KeyPad.*
- `void gslc_ElemXKeyPadCfgSetRoundEn (gslc_tsXKeyPadCfg *pConfig, bool bEn)`  
*Update the KeyPad configuration to enable rounded button corners.*
- `void gslc_ElemXKeyPadCfgSetButtonSz (gslc_tsXKeyPadCfg *pConfig, int8_t nButtonSzW, int8_t nButton↵  
SzH)`  
*Update the KeyPad configuration to define the KeyPad button sizing.*
- `void gslc_ElemXKeyPadCfgSetButtonSpace (gslc_tsXKeyPadCfg *pConfig, int8_t nSpaceX, int8_t nSpaceY)`  
*Update the KeyPad configuration to define the KeyPad button spacing.*
- `void gslc_XKeyPadDrawVirtualTxt (gslc_tsGui *pGui, gslc_tsRect rElem, gslc_tsXKeyPad *pKeyPad, gslc↵  
_tsColor cColFrame, gslc_tsColor cColFill, gslc_tsColor cColTxt)`  
*Draw a virtual Text Element.*
- `void gslc_XKeyPadDrawVirtualBtn (gslc_tsGui *pGui, gslc_tsRect rElem, char *pStrBuf, uint8_t nStrBuf↵  
Max, int16_t nFontId, gslc_tsColor cColFrame, gslc_tsColor cColFill, gslc_tsColor cColFillGlow, gslc_tsColor  
cColTxt, bool bRoundedEn, bool bGlow, bool bFocus)`  
*Draw a virtual textual Button Element.*
- `void gslc_ElemXKeyPadInputAsk (gslc_tsGui *pGui, gslc_tsElemRef *pKeyPadRef, int16_t nPgPopup,  
gslc_tsElemRef *pTxtRef)`  
*Trigger a KeyPad popup and associate it with a text element.*
- `char * gslc_ElemXKeyPadInputGet (gslc_tsGui *pGui, gslc_tsElemRef *pTxtRef, void *pvCbData)`  
*Complete a KeyPad popup by retrieving the input data and storing it in the text element.*

## Variables

- static const int `RBIT_TXT`
- static const int `RBIT_KEYONE`
- static const int `RBIT_KEYALL`
- static const int `RBIT_CTRL`

## 9.11.1 Macro Definition Documentation

#### 9.11.1.1 DEBUG\_XKEYPAD

```
#define DEBUG_XKEYPAD
```

Debug message for XKeyPad (1=enabled, 0=disabled)

#### 9.11.1.2 GSLC\_TYPEX\_KEYPAD

```
#define GSLC_TYPEX_KEYPAD
```

#### 9.11.1.3 XKEYPAD\_BUF\_MAX

```
#define XKEYPAD_BUF_MAX
```

#### 9.11.1.4 XKEYPAD\_CB\_STATE\_CANCEL

```
#define XKEYPAD_CB_STATE_CANCEL
```

#### 9.11.1.5 XKEYPAD\_CB\_STATE\_DONE

```
#define XKEYPAD_CB_STATE_DONE
```

#### 9.11.1.6 XKEYPAD\_CB\_STATE\_UPDATE

```
#define XKEYPAD_CB_STATE_UPDATE
```

#### 9.11.1.7 XKEYPAD\_CURSOR\_CH

```
#define XKEYPAD_CURSOR_CH
```

#### 9.11.1.8 XKEYPAD\_CURSOR\_ENHANCED

```
#define XKEYPAD_CURSOR_ENHANCED
```

#### 9.11.1.9 XKEYPAD\_KEY\_LEN

```
#define XKEYPAD_KEY_LEN
```

#### 9.11.1.10 XKEYPAD\_REDRAW\_ALL

```
#define XKEYPAD_REDRAW_ALL
```

#### 9.11.1.11 XKEYPAD\_REDRAW\_FULL

```
#define XKEYPAD_REDRAW_FULL
```

#### 9.11.1.12 XKEYPAD\_REDRAW\_KEY

```
#define XKEYPAD_REDRAW_KEY
```

#### 9.11.1.13 XKEYPAD\_REDRAW\_NONE

```
#define XKEYPAD_REDRAW_NONE
```

#### 9.11.1.14 XKEYPAD\_REDRAW\_TXT

```
#define XKEYPAD_REDRAW_TXT
```

### 9.11.2 Typedef Documentation



#### 9.11.2.1 GSLC\_CB\_XKEYPAD\_BTN\_EVT

```
typedef void(* GSLC_CB_XKEYPAD_BTN_EVT) (void *pvKeyPad, uint8_t nId, gslc\_tsXKeyPadResult
*psResult)
```

#### 9.11.2.2 GSLC\_CB\_XKEYPAD\_LABEL\_GET

```
typedef void(* GSLC_CB_XKEYPAD_LABEL_GET) (void *pvKeyPad, uint8_t nId, uint8_t nStrMax, char
*pStr)
```

#### 9.11.2.3 GSLC\_CB\_XKEYPAD\_RESET

```
typedef void(* GSLC_CB_XKEYPAD_RESET) (void *pvKeyPadConfig)
```

#### 9.11.2.4 GSLC\_CB\_XKEYPAD\_SYTLE\_GET

```
typedef void(* GSLC_CB_XKEYPAD_SYTLE_GET) (void *pvKeyPad, uint8_t nId, bool *bVisible, gslc\_↵
tsColor *pcolTxt, gslc\_tsColor *pcolFrame, gslc\_tsColor *pcolFill, gslc\_tsColor *pcolGlow)
```

#### 9.11.2.5 GSLC\_CB\_XKEYPAD\_TXT\_INIT

```
typedef void(* GSLC_CB_XKEYPAD_TXT_INIT) (void *pvKeyPad)
```

#### 9.11.2.6 [gslc\\_tsKey](#)

```
typedef struct gslc\_tsKey gslc\_tsKey
```

Key information. Defines everything we need to know about a particular key.

### 9.11.3 Enumeration Type Documentation

#### 9.11.3.1 anonymous enum

```
anonymous enum
```

**Enumerator**

KEYPAD_ID_BACKSPACE	
KEYPAD_ID_SPACE	
KEYPAD_ID_ESC	
KEYPAD_ID_ENTER	
KEYPAD_ID_SWAP_PAD	
KEYPAD_ID_SCROLL_LEFT	
KEYPAD_ID_SCROLL_RIGHT	
KEYPAD_ID_BASIC_START	
KEYPAD_ID_TXT	
KEYPAD_ID__END	

**9.11.3.2 anonymous enum**

anonymous enum

**Enumerator**

E_XKEYPAD_TYPE_BASIC	
E_XKEYPAD_TYPE_SPECIAL	
E_XKEYPAD_TYPE_TXT	
E_XKEYPAD_TYPE_UNUSED	
E_XKEYPAD_TYPE_END	

**9.11.3.3 gslc\_tsXKeyPadAttrib**

enum `gslc_tsXKeyPadAttrib`

**Enumerator**

E_XKEYPAD_ATTRIB_FOCUS	
E_XKEYPAD_ATTRIB_GLOW	

**9.11.4 Function Documentation****9.11.4.1 gslc\_ElemXKeyPadCfgInit()**

```
void gslc_ElemXKeyPadCfgInit (
    gslc_tsXKeyPadCfg * pConfig )
```

Provide default initialization for the base XKeyPad.

- These defaults will be overwritten by variant-specific initialization or through user configuration APIs.

#### Parameters

in	<i>pConfig</i>	Ptr to the KeyPad base config structure
----	----------------	---

#### Returns

none

#### 9.11.4.2 gslc\_ElemXKeyPadCfgSetButtonSpace()

```
void gslc_ElemXKeyPadCfgSetButtonSpace (
    gslc_tsXKeyPadCfg * pConfig,
    int8_t nSpaceX,
    int8_t nSpaceY )
```

Update the KeyPad configuration to define the KeyPad button spacing.

- This defines the inset amount (X and Y) from the Button Size
- A spacing of (1,1) will mean that the button is drawn with a 1 pixel margin around the grid defined by the Button Size

#### Parameters

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>nSpaceX</i>	Amount to inset button in horizontal direction (pixels)
in	<i>nSpaceY</i>	Amount to inset button in vertical direction (pixels)

#### Returns

none

#### 9.11.4.3 gslc\_ElemXKeyPadCfgSetButtonSz()

```
void gslc_ElemXKeyPadCfgSetButtonSz (
    gslc_tsXKeyPadCfg * pConfig,
    int8_t nButtonSzW,
    int8_t nButtonSzH )
```

Update the KeyPad configuration to define the KeyPad button sizing.

**Parameters**

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>nButtonSzW</i>	Width of buttons in pixels
in	<i>nButtonSzH</i>	Height of buttons in pixels

**Returns**

none

**9.11.4.4 gslc\_ElemXKeyPadCfgSetRoundEn()**

```
void gslc_ElemXKeyPadCfgSetRoundEn (
    gslc_tsXKeyPadCfg * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable rounded button corners.

**Parameters**

in, out	<i>pConfig</i>	Pointer to the XKeyPad base config structure
in	<i>bEn</i>	Enable rounded corners

**Returns**

none

**9.11.4.5 gslc\_ElemXKeyPadDataTargetIdGet()**

```
int16_t gslc_ElemXKeyPadDataTargetIdGet (
    gslc_tsGui * pGui,
    void * pvData )
```

Fetch the element target ID associated with this KeyPad.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pvData</i>	: Void ptr to callback data structure

**Returns**

Target Element ID or GSLC\_ID\_NONE if unspecified

## 9.11.4.6 gslc\_ElemXKeyPadDataValGet()

```
char* gslc_ElemXKeyPadDataValGet (
    gslc_tsGui * pGui,
    void * pvData )
```

Fetch the final value string of the KeyPad from a callback.

## Parameters

in	<i>pGui</i>	Pointer to GUI
out	<i>pvData</i>	: Void ptr to callback data structure

## Returns

Pointer to edited character string

## 9.11.4.7 gslc\_ElemXKeyPadInputAsk()

```
void gslc_ElemXKeyPadInputAsk (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pKeyPadRef,
    int16_t nPgPopup,
    gslc_tsElemRef * pTxtRef )
```

Trigger a KeyPad popup and associate it with a text element.

- This function also updates the maximum KeyPad buffer length to match that of the target text element, up to the maximum XKEYPAD\_BUF\_MAX.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pKeyPadRef</i>	Pointer to KeyPad element reference
in	<i>nPgPopup</i>	Page enum that contains the popup to show
in	<i>pTxtRef</i>	Pointer to associated text field element reference

## Returns

none

## 9.11.4.8 gslc\_ElemXKeyPadInputGet()

```
char* gslc_ElemXKeyPadInputGet (
    gslc_tsGui * pGui,
```

```
gslc_tsElemRef * pTxtRef,
void * pvCbData )
```

Complete a KeyPad popup by retrieving the input data and storing it in the text element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pTxtRef</i>	Pointer to associated text field element reference
in	<i>pvCbData</i>	Void pointer to callback function's pvData

#### Returns

The text string that was fetched from the KeyPad (NULL terminated)

#### 9.11.4.9 gslc\_ElemXKeyPadTargetRefSet()

```
void gslc_ElemXKeyPadTargetRefSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsElemRef * pTargetRef )
```

Set target element reference for KeyPad return value.

- The Target Reference is used in the GSLC\_CB\_INPUT callback so that the user has the context needed to determine which field should be edited with the contents of the KeyPad edit field
- It is expected that the user will call this function when showing the KeyPad popup dialog

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
in	<i>pTargetRef</i>	Element reference for target of KeyPad value

#### Returns

none

#### 9.11.4.10 gslc\_ElemXKeyPadValGet()

```
bool gslc_ElemXKeyPadValGet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
```

```
char * pStrBuf,  
uint8_t nStrBufMax )
```

Fetch the current value string associated with KeyPad element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
out	<i>pStrBuf</i>	String buffer
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf) including terminator

**Returns**

true if success, false otherwise

**9.11.4.11 gslc\_ElemXKeyPadValSet()**

```
void gslc_ElemXKeyPadValSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    const char * pStrBuf )
```

Set the current value for the editable text field.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to KeyPad Element reference
in	<i>pStrBuf</i>	String to copy into keypad

**Returns**

none

**9.11.4.12 gslc\_ElemXKeyPadValSetCb()**

```
void gslc_ElemXKeyPadValSetCb (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_INPUT pfuncCb )
```

Set the callback function associated with the KeyPad.

- This function will be called during updates and OK / Cancel

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element Reference for KeyPad
in	<i>pfuncCb</i>	Callback function pointer



**Returns**

none

**9.11.4.13 gslc\_XKeyPadCreateBase()**

```

gslc_tsElemRef* gslc_XKeyPadCreateBase (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pXData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg * pConfig )

```

Create a KeyPad Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Pointer to base Config options

**Returns**

Pointer to Element or NULL if failure

**9.11.4.14 gslc\_XKeyPadDraw()**

```

bool gslc_XKeyPadDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a KeyPad element on the screen.

- Called during redraw

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.11.4.15 `gslc_XKeyPadDrawKey()`**

```
void gslc_XKeyPadDrawKey (
    gslc_tsGui * pGui,
    gslc_tsXKeyPad * pXData,
    gslc_tsKey * pKey,
    bool bGlow,
    bool bFocus )
```

Draw a key to the screen.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>pKey</i>	Ptr to key being drawn
in	<i>bGlow</i>	Indicate if key is in glow state
in	<i>bFocus</i>	Indicate if key is in focus state

**Returns**

none

**9.11.4.16 `gslc_XKeyPadDrawVirtualBtn()`**

```
void gslc_XKeyPadDrawVirtualBtn (
    gslc_tsGui * pGui,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId,
    gslc_tsColor cColFrame,
    gslc_tsColor cColFill,
    gslc_tsColor cColFillGlow,
    gslc_tsColor cColTxt,
```

```

    bool bRoundedEn,
    bool bGlow,
    bool bFocus )

```

Draw a virtual textual Button Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pStrBuf</i>	String to copy into element
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf).
in	<i>nFontId</i>	Font ID to use for text display
in	<i>cColFrame</i>	Frame color for element
in	<i>cColFill</i>	Fill color for element
in	<i>cColFillGlow</i>	Fill color for element when glowing/focused
in	<i>cColTxt</i>	Text color for element
in	<i>bRoundedEn</i>	Use Rounded Corners?
in	<i>bGlow</i>	Indicate btn is in glow state
in	<i>bFocus</i>	Indicate btn is in focus state

#### Returns

none

#### 9.11.4.17 gslc\_XKeyPadDrawVirtualTxt()

```

void gslc_XKeyPadDrawVirtualTxt (
    gslc_tsGui * pGui,
    gslc_tsRect rElem,
    gslc_tsXKeyPad * pKeyPad,
    gslc_tsColor cColFrame,
    gslc_tsColor cColFill,
    gslc_tsColor cColTxt )

```

Draw a virtual Text Element.

- Creates a text string with filled background

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pKeyPad</i>	Pointer to KeyPad struct
in	<i>cColFrame</i>	Frame color for element
in	<i>cColFill</i>	Fill color for element
in	<i>cColTxt</i>	Text color for element

**Returns**

none

**9.11.4.18 gslc\_XKeyPadFocusSetDefault()**

```
void gslc_XKeyPadFocusSetDefault (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

**Todo** Doc

**9.11.4.19 gslc\_XKeyPadLayoutSet()**

```
bool gslc_XKeyPadLayoutSet (
    gslc_tsXKeyPadCfg * pConfig,
    int8_t eLayoutSel )
```

Select a new KeyPad layout.

- Multiple KeyPad layouts can share the same key key definition array (eg. KEYPAD\_LAYOUT)
- This function returns an indication of whether a full KeyPad control redraw is required, ie. the KeyPad layout definition has changed. With changes in the KeyPad definition, there may be a different number of visible keys or arrangements, necessitating a background redraw.

**Parameters**

in	<i>pConfig</i>	Ptr to the KeyPad configuration
in	<i>eLayoutSel</i>	Layout index to select

**Returns**

true if a full redraw should be performed

**9.11.4.20 gslc\_XKeyPadLookupId()**

```
int16_t gslc_XKeyPadLookupId (
    gslc_tsKey * pKeys,
    uint8_t nKeyId )
```

Find a key ID within a KeyPad label array and return it's index into the array.

- It is expected that the KeyPad label array is terminated with KEYPAD\_ID\_\_END

## Parameters

in	<i>pKeys</i>	Ptr to the Keypad label array
in	<i>nKeyId</i>	Key ID to look for

## Returns

the index into the array if the ID was found or -1 if the key ID was not found

## 9.11.4.21 gslc\_XKeyPadLookupSpecialId()

```
int16_t gslc_XKeyPadLookupSpecialId (
    gslc_tsLabelSpecial * pLabels,
    uint8_t nKeyId )
```

Find a key ID within a Keypad special label array and return it's index into the array.

- It is expected that the Keypad label array is terminated with KEYPAD\_ID\_\_END

## Parameters

in	<i>pLabels</i>	Ptr to the Keypad special label array
in	<i>nKeyId</i>	Key ID to look for

## Returns

the index into the array if the ID was found or -1 if the key ID was not found

## 9.11.4.22 gslc\_XKeyPadPendRedrawAddKey()

```
void gslc_XKeyPadPendRedrawAddKey (
    gslc_tsXKeyPadResult * pResult,
    int16_t nId )
```

**Todo** Doc

#### 9.11.4.23 `gslc_XKeyPadPendRedrawAddTxt()`

```
void gslc_XKeyPadPendRedrawAddTxt (
    gslc_tsXKeyPadResult * pResult )
```

**Todo** Doc

#### 9.11.4.24 `gslc_XKeyPadPendRedrawReset()`

```
void gslc_XKeyPadPendRedrawReset (
    gslc_tsXKeyPadResult * pResult )
```

**Todo** Doc

#### 9.11.4.25 `gslc_XKeyPadRedrawUpdate()`

```
void gslc_XKeyPadRedrawUpdate (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

**Todo** Doc

#### 9.11.4.26 `gslc_XKeyPadSizeAllGet()`

```
void gslc_XKeyPadSizeAllGet (
    gslc_tsKey ** pLayouts,
    uint8_t nNumLayouts,
    uint8_t * pnRows,
    uint8_t * pnCols )
```

Calculate the overall dimensions of the KeyPad control encompassing all available layouts for the KeyPad, leveraging the computation in `gslc_XKeyPadSizeGet()`.

##### Parameters

in	<i>pLayouts</i>	Ptr to the array of KeyPad layouts
in	<i>nNumLayouts</i>	Number of layouts in pLayouts
out	<i>pnRows</i>	Ptr for the number of rows
out	<i>pnCols</i>	Ptr for the number of columns

## Returns

none

## 9.11.4.27 gslc\_XKeyPadSizeGet()

```
void gslc_XKeyPadSizeGet (
    gslc_tsKey * pLayout,
    uint8_t * pnRows,
    uint8_t * pnCols,
    int8_t * pnIndFirst,
    int8_t * pnIndLast )
```

Calculate the overall dimensions of the KeyPad control encompassing the text field and key buttons.

The dimension is calculated in units of the configured key size (width and height), and accounts for any column spans. It also returns the index of the first and last keys on the keypad.

## Parameters

in	<i>pLayout</i>	Ptr to the KeyPad layout
out	<i>pnRows</i>	Ptr for the number of rows
out	<i>pnCols</i>	Ptr for the number of columns
out	<i>pnIndFirst</i>	Ptr for the index of first key
out	<i>pnIndLast</i>	Ptr for the index of last key

## Returns

none

## 9.11.4.28 gslc\_XKeyPadTouch()

```
bool gslc_XKeyPadTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch (up,down,move) events to KeyPad element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

**Returns**

true if success, false otherwise

**9.11.4.29 gslc\_XKeyPadTrackSet()**

```
void gslc_XKeyPadTrackSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nInd,
    gslc_tsXKeyPadAttrib eAttrib )
```

**Todo** Doc

**9.11.4.30 gslc\_XKeyPadTxtAddCh()**

```
bool gslc_XKeyPadTxtAddCh (
    gslc_tsXKeyPad * pKeyPad,
    char ch,
    uint8_t nPos )
```

Add a character to the KeyPad text field at the specified offset (nPos).

Providing an offset equal to the end of the existing buffer length will cause the addition to the end, whereas an offset within the buffer will cause an insert.

- An addition that causes the buffer length to exceed the maximum allowed will result in the end of the resulting buffer to be truncated.
- Typically the addition will be done at the current cursor position.
- If the insertion is ahead of the cursor, then the cursor position may be increased.

**Parameters**

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>ch</i>	Character to add
in	<i>nPos</i>	Buffer position for the insertion

**Returns**

true if the text field should be redrawn, false if no redraw is needed (ie. no change)



#### 9.11.4.31 gslc\_XKeyPadTxtAddStr()

```
bool gslc_XKeyPadTxtAddStr (
    gslc_tsXKeyPad * pKeyPad,
    const char * pStr,
    uint8_t nPos )
```

Add a string to the KeyPad text field at the specified offset (nPos).

Providing an offset equal to the end of the existing buffer length will cause the addition to the end, whereas an offset within the buffer will cause an insert.

- An addition that causes the buffer length to exceed the maximum allowed will result in the end of the resulting buffer to be truncated.
- Typically the addition will be done at the current cursor position.
- If the insertion is ahead of the cursor, then the cursor position may be increased.
- This routine may be useful when adding multi-byte characters for future support of foreign characters.

##### Parameters

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>pStr</i>	String to add
in	<i>nPos</i>	Buffer position for the insertion

##### Returns

true if the text field should be redrawn, false if no redraw is needed (ie. no change)

#### 9.11.4.32 gslc\_XKeyPadTxtDelCh()

```
bool gslc_XKeyPadTxtDelCh (
    gslc_tsXKeyPad * pKeyPad,
    uint8_t nPos )
```

Remove a character from the KeyPad text field at the specified offset (nPos).

- Typically the addition will be done at the current cursor position.
- If the removal is ahead of the cursor, then the cursor position may be decreased.

##### Parameters

in	<i>pKeyPad</i>	Ptr to the KeyPad
in	<i>nPos</i>	Buffer position for the removal

**Returns**

true if the text field should be redrawn, false if no redraw is needed (ie. no change)

**9.11.5 Variable Documentation****9.11.5.1 RBIT\_CTRL**

```
const int RBIT_CTRL [static]
```

**9.11.5.2 RBIT\_KEYALL**

```
const int RBIT_KEYALL [static]
```

**9.11.5.3 RBIT\_KEYONE**

```
const int RBIT_KEYONE [static]
```

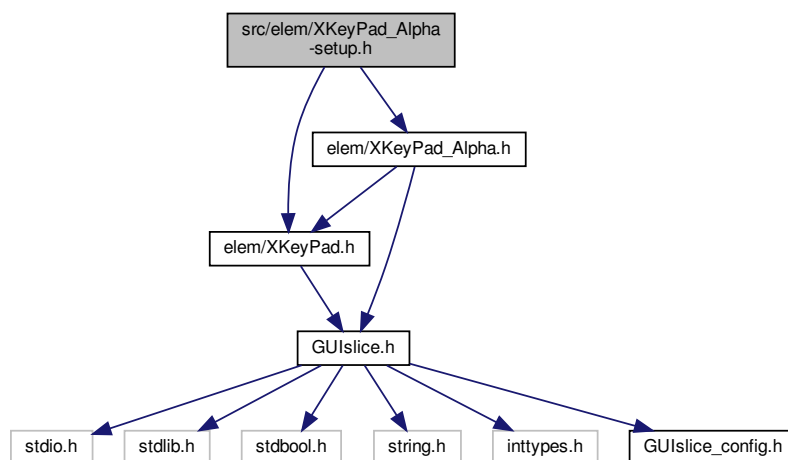
**9.11.5.4 RBIT\_TXT**

```
const int RBIT_TXT [static]
```

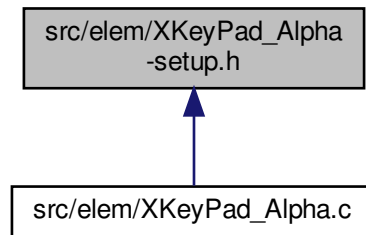
**9.12 src/elem/XKeyPad\_Alpha-setup.h File Reference**

```
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Alpha.h"
```

Include dependency graph for XKeyPad\_Alpha-setup.h:



This graph shows which files directly or indirectly include this file:



## Macros

- #define [XKEYPAD\\_EXTEND\\_CHAR](#)
- #define [XKEYPAD\\_LABEL\\_MAX](#)
- #define [XKEYPAD\\_DISP\\_MAX](#)
- #define [XKEYPAD\\_KEY\\_W](#)
- #define [XKEYPAD\\_KEY\\_H](#)
- #define [XKEYPAD\\_SPACING\\_X](#)
- #define [XKEYPAD\\_SPACING\\_Y](#)
- #define [XKEYPAD\\_COL\\_DISABLE\\_TXT](#)
- #define [XKEYPAD\\_COL\\_DISABLE\\_FILL](#)
- #define [XKEYPAD\\_COL\\_DEF\\_TXT](#)
- #define [XKEYPAD\\_COL\\_DEF\\_FRAME](#)
- #define [XKEYPAD\\_COL\\_DEF\\_FILL](#)
- #define [XKEYPAD\\_COL\\_DEF\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_BASIC\\_FILL](#)
- #define [XKEYPAD\\_COL\\_BASIC\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_TEXT\\_TXT](#)
- #define [XKEYPAD\\_COL\\_TEXT\\_FILL](#)
- #define [XKEYPAD\\_COL\\_TEXT\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_SPACE\\_FILL](#)
- #define [XKEYPAD\\_COL\\_SPACE\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_ESC\\_FILL](#)
- #define [XKEYPAD\\_COL\\_ESC\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_ENTER\\_FILL](#)
- #define [XKEYPAD\\_COL\\_ENTER\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_SCROLL\\_L\\_FILL](#)
- #define [XKEYPAD\\_COL\\_SCROLL\\_L\\_GLOW](#)
- #define [XKEYPAD\\_COL\\_SCROLL\\_R\\_FILL](#)
- #define [XKEYPAD\\_COL\\_SCROLL\\_R\\_GLOW](#)

## Enumerations

- enum [gslc\\_teXKeyPadSel](#) {  
[E\\_XKEYPAD\\_SET\\_UPPER](#), [E\\_XKEYPAD\\_SET\\_LOWER](#), [E\\_XKEYPAD\\_SET\\_NUM](#), [E\\_XKEYPAD\\_SET\\_MAX](#),  
[E\\_XKEYPAD\\_SET\\_NUM](#), [E\\_XKEYPAD\\_SET\\_MAX](#) }

## Variables

- static const char \* [XKEYPAD\\_LABEL\\_SPACE](#)
- static const int8\_t [XKEYPAD\\_LAYOUT\\_DEFAULT](#)
- static [gslc\\_tsLabelSpecial](#) [KEYPAD\\_SPECIAL\\_LABEL](#) []
- static const char \* [KEYPAD\\_SET\\_LABEL](#) []
- static const char \* [KEYPAD\\_SPECIAL\\_SELECT](#) []
- static [gslc\\_tsKey](#) [KEYPAD\\_LAYOUT](#) []
- static [gslc\\_tsKey](#) \* [KEYPAD\\_LAYOUTS](#) [[E\\_XKEYPAD\\_SET\\_\\_MAX](#)]

## 9.12.1 Macro Definition Documentation

### 9.12.1.1 XKEYPAD\_COL\_BASIC\_FILL

```
#define XKEYPAD_COL_BASIC_FILL
```

### 9.12.1.2 XKEYPAD\_COL\_BASIC\_GLOW

```
#define XKEYPAD_COL_BASIC_GLOW
```

### 9.12.1.3 XKEYPAD\_COL\_DEF\_FILL

```
#define XKEYPAD_COL_DEF_FILL
```

### 9.12.1.4 XKEYPAD\_COL\_DEF\_FRAME

```
#define XKEYPAD_COL_DEF_FRAME
```

### 9.12.1.5 XKEYPAD\_COL\_DEF\_GLOW

```
#define XKEYPAD_COL_DEF_GLOW
```

#### 9.12.1.6 XKEYPAD\_COL\_DEF\_TXT

```
#define XKEYPAD_COL_DEF_TXT
```

#### 9.12.1.7 XKEYPAD\_COL\_DISABLE\_FILL

```
#define XKEYPAD_COL_DISABLE_FILL
```

#### 9.12.1.8 XKEYPAD\_COL\_DISABLE\_TXT

```
#define XKEYPAD_COL_DISABLE_TXT
```

#### 9.12.1.9 XKEYPAD\_COL\_ENTER\_FILL

```
#define XKEYPAD_COL_ENTER_FILL
```

#### 9.12.1.10 XKEYPAD\_COL\_ENTER\_GLOW

```
#define XKEYPAD_COL_ENTER_GLOW
```

#### 9.12.1.11 XKEYPAD\_COL\_ESC\_FILL

```
#define XKEYPAD_COL_ESC_FILL
```

#### 9.12.1.12 XKEYPAD\_COL\_ESC\_GLOW

```
#define XKEYPAD_COL_ESC_GLOW
```

#### 9.12.1.13 XKEYPAD\_COL\_SCROLL\_L\_FILL

```
#define XKEYPAD_COL_SCROLL_L_FILL
```

**9.12.1.14 XKEYPAD\_COL\_SCROLL\_L\_GLOW**

```
#define XKEYPAD_COL_SCROLL_L_GLOW
```

**9.12.1.15 XKEYPAD\_COL\_SCROLL\_R\_FILL**

```
#define XKEYPAD_COL_SCROLL_R_FILL
```

**9.12.1.16 XKEYPAD\_COL\_SCROLL\_R\_GLOW**

```
#define XKEYPAD_COL_SCROLL_R_GLOW
```

**9.12.1.17 XKEYPAD\_COL\_SPACE\_FILL**

```
#define XKEYPAD_COL_SPACE_FILL
```

**9.12.1.18 XKEYPAD\_COL\_SPACE\_GLOW**

```
#define XKEYPAD_COL_SPACE_GLOW
```

**9.12.1.19 XKEYPAD\_COL\_TEXT\_FILL**

```
#define XKEYPAD_COL_TEXT_FILL
```

**9.12.1.20 XKEYPAD\_COL\_TEXT\_GLOW**

```
#define XKEYPAD_COL_TEXT_GLOW
```

**9.12.1.21 XKEYPAD\_COL\_TEXT\_TXT**

```
#define XKEYPAD_COL_TEXT_TXT
```

#### 9.12.1.22 XKEYPAD\_DISP\_MAX

```
#define XKEYPAD_DISP_MAX
```

#### 9.12.1.23 XKEYPAD\_EXTEND\_CHAR

```
#define XKEYPAD_EXTEND_CHAR
```

#### 9.12.1.24 XKEYPAD\_KEY\_H

```
#define XKEYPAD_KEY_H
```

#### 9.12.1.25 XKEYPAD\_KEY\_W

```
#define XKEYPAD_KEY_W
```

#### 9.12.1.26 XKEYPAD\_LABEL\_MAX

```
#define XKEYPAD_LABEL_MAX
```

#### 9.12.1.27 XKEYPAD\_SPACING\_X

```
#define XKEYPAD_SPACING_X
```

#### 9.12.1.28 XKEYPAD\_SPACING\_Y

```
#define XKEYPAD_SPACING_Y
```

### 9.12.2 Enumeration Type Documentation

#### 9.12.2.1 gslc\_teXKeyPadSel

```
enum gslc\_teXKeyPadSel
```

## Enumerator

E_XKEYPAD_SET_UPPER	
E_XKEYPAD_SET_LOWER	
E_XKEYPAD_SET_NUM	
E_XKEYPAD_SET__MAX	
E_XKEYPAD_SET_NUM	
E_XKEYPAD_SET__MAX	

### 9.12.3 Variable Documentation

#### 9.12.3.1 KEYPAD\_LAYOUT

```
gslc_tsKey KEYPAD_LAYOUT[] [static]
```

#### 9.12.3.2 KEYPAD\_LAYOUTS

```
gslc_tsKey* KEYPAD_LAYOUTS[E_XKEYPAD_SET__MAX] [static]
```

#### 9.12.3.3 KEYPAD\_SET\_LABEL

```
const char* KEYPAD_SET_LABEL[] [static]
```

#### 9.12.3.4 KEYPAD\_SPECIAL\_LABEL

```
gslc_tsLabelSpecial KEYPAD_SPECIAL_LABEL[] [static]
```

#### 9.12.3.5 KEYPAD\_SPECIAL\_SELECT

```
const char* KEYPAD_SPECIAL_SELECT[] [static]
```



## 9.12.3.6 XKEYPAD\_LABEL\_SPACE

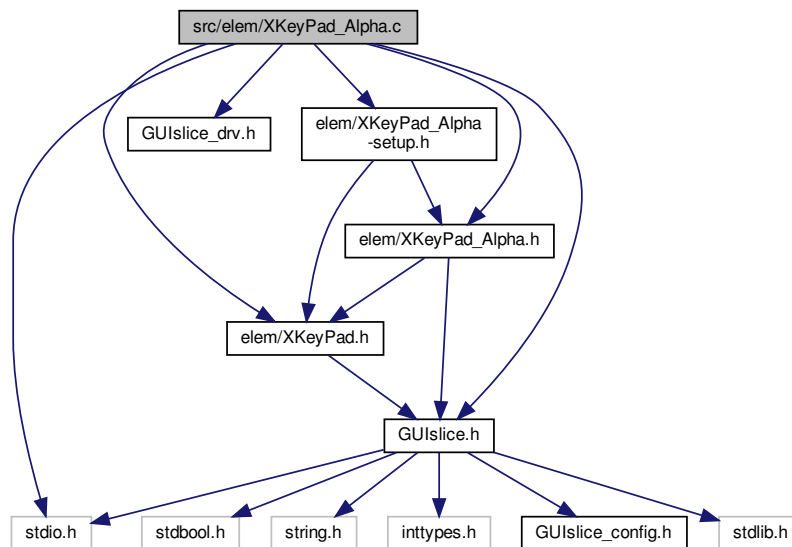
```
const char* XKEYPAD_LABEL_SPACE [static]
```

## 9.12.3.7 XKEYPAD\_LAYOUT\_DEFAULT

```
const int8_t XKEYPAD_LAYOUT_DEFAULT [static]
```

## 9.13 src/elem/XKeyPad\_Alpha.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Alpha.h"
#include <stdio.h>
#include "elem/XKeyPad_Alpha-setup.h"
Include dependency graph for XKeyPad_Alpha.c:
```



## Functions

- void [gslc\\_ElemXKeyPadReset\\_Alpha](#) (void \*pvConfig)  
*Callback function to reset internal state.*
- void [gslc\\_ElemXKeyPadTxtInit\\_Alpha](#) (void \*pvKeyPad)  
*Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).*
- void [gslc\\_ElemXKeyPadLabelGet\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, uint8\_t nStrMax, char \*pStr)  
*Callback function to retrieve the label associated with a KeyPad button.*

- void [gslc\\_ElemXKeyPadStyleGet\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, bool \*pbVisible, [gslc\\_tsColor](#) \*pcol←→Txt, [gslc\\_tsColor](#) \*pcolFrame, [gslc\\_tsColor](#) \*pcolFill, [gslc\\_tsColor](#) \*pcolGlow)  
*Callback function to retrieve the style associated with a KeyPad button.*
- void [gslc\\_ElemXKeyPadBtnEvt\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, [gslc\\_tsXKeyPadResult](#) \*psResult)  
*Callback function activated when a key has been pressed.*
- [gslc\\_tsXKeyPadCfg\\_Alpha](#) [gslc\\_ElemXKeyPadCfgInit\\_Alpha](#) ()  
*Initialize the KeyPad config structure.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXKeyPadCreate\\_Alpha](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXKeyPad](#) \*pXData, int16\_t nX0, int16\_t nY0, int8\_t nFontId, [gslc\\_tsXKeyPadCfg\\_Alpha](#) \*pConfig)  
*Create a KeyPad Element.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.13.1 Function Documentation

### 9.13.1.1 [gslc\\_ElemXKeyPadBtnEvt\\_Alpha\(\)](#)

```
void gslc_ElemXKeyPadBtnEvt_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    gslc\_tsXKeyPadResult * psResult )
```

Callback function activated when a key has been pressed.

This callback is used to enable the KeyPad variant to handle any events associated with the key press and update any internal state.

- The callback is also used to determine whether any redraw actions need to be taken.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>psResult</i>	The returned state vector (including redraw)

#### Returns

none

## 9.13.1.2 gslc\_ElemXKeyPadCfgInit\_Alpha()

```
gslc_tsXKeyPadCfg_Alpha gslc_ElemXKeyPadCfgInit_Alpha ( )
```

Initialize the KeyPad config structure.

- This routine should be called to initialize the configuration data structure before calling any of the KeyPad config APIs

## Returns

Initialized KeyPad config structure

## 9.13.1.3 gslc\_ElemXKeyPadCreate\_Alpha()

```
gslc_tsElemRef* gslc_ElemXKeyPadCreate_Alpha (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pXData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg_Alpha * pConfig )
```

Create a KeyPad Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Ptr to config options

## Returns

Pointer to Element or NULL if failure

#### 9.13.1.4 gslc\_ElemXKeyPadLabelGet\_Alpha()

```
void gslc_ElemXKeyPadLabelGet_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    uint8_t nStrMax,
    char * pStr )
```

Callback function to retrieve the label associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

##### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
in	<i>nStrMax</i>	Maximum length of return string (including NULL)
out	<i>pStr</i>	Buffer for the returned label

##### Returns

none

#### 9.13.1.5 gslc\_ElemXKeyPadReset\_Alpha()

```
void gslc_ElemXKeyPadReset_Alpha (
    void * pvConfig )
```

Callback function to reset internal state.

##### Parameters

in	<i>pvConfig</i>	Void ptr to the KeyPad config
----	-----------------	-------------------------------

##### Returns

none

#### 9.13.1.6 gslc\_ElemXKeyPadStyleGet\_Alpha()

```
void gslc_ElemXKeyPadStyleGet_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    bool * pbVisible,
    gslc_tsColor * pColTxt,
```

```

gslc_tsColor * pcolFrame,
gslc_tsColor * pcolFill,
gslc_tsColor * pcolGlow )

```

Callback function to retrieve the style associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

- This function is used to assign the color and visibility state of the keys at runtime.
- This function can also be used to change the appearance dynamically, according to internal state (eg. dimmed buttons).

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>pbVisible</i>	The returned visibility state
out	<i>pcolTxt</i>	The returned text color
out	<i>pcolFrame</i>	The returned key's frame color
out	<i>pcolFill</i>	The returned key's fill color
out	<i>pcolGlow</i>	The returned key's fill color when highlighted

#### Returns

none

#### 9.13.1.7 gslc\_ElemXKeyPadTxtInit\_Alpha()

```

void gslc_ElemXKeyPadTxtInit_Alpha (
    void * pvKeyPad )

```

Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).

- This is used to ensure any KeyPad variant state can be kept in sync with the text string.
- For example, if a numeric KeyPad is initiaized with a string that contains a minus sign, an internal negation flag might be set.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
----	-----------------	------------------------

#### Returns

none

### 9.13.2 Variable Documentation

#### 9.13.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

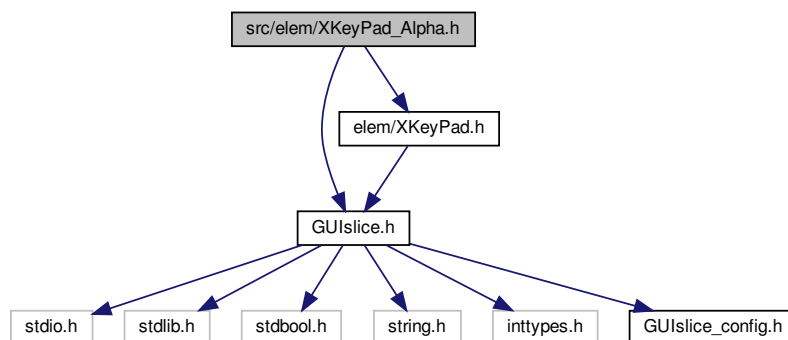
#### 9.13.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

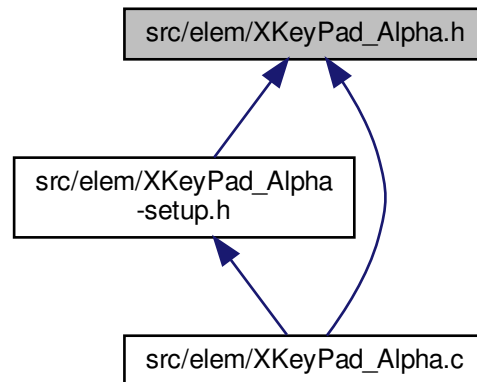
## 9.14 src/elem/XKeyPad\_Alpha.h File Reference

```
#include "GUIslice.h"  
#include "elem/XKeyPad.h"
```

Include dependency graph for XKeyPad\_Alpha.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXKeyPadCfg\\_Alpha](#)

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXKeyPadCreate\\_Alpha](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXKeyPad](#) \*pXData, int16\_t nX0, int16\_t nY0, int8\_t nFontId, [gslc\\_tsXKeyPadCfg\\_Alpha](#) \*pConfig)  
*Create a KeyPad Element.*
- [gslc\\_tsXKeyPadCfg\\_Alpha](#) [gslc\\_ElemXKeyPadCfgInit\\_Alpha](#) ()  
*Initialize the KeyPad config structure.*
- void [gslc\\_ElemXKeyPadReset\\_Alpha](#) (void \*pvConfig)  
*Callback function to reset internal state.*
- void [gslc\\_ElemXKeyPadTxtInit\\_Alpha](#) (void \*pvKeyPad)  
*Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).*
- void [gslc\\_ElemXKeyPadLabelGet\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, uint8\_t nStrMax, char \*pStr)  
*Callback function to retrieve the label associated with a KeyPad button.*
- void [gslc\\_ElemXKeyPadStyleGet\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, bool \*pbVisible, [gslc\\_tsColor](#) \*pcol↔Txt, [gslc\\_tsColor](#) \*pcolFrame, [gslc\\_tsColor](#) \*pcolFill, [gslc\\_tsColor](#) \*pcolGlow)  
*Callback function to retrieve the style associated with a KeyPad button.*
- void [gslc\\_ElemXKeyPadBtnEvt\\_Alpha](#) (void \*pvKeyPad, uint8\_t nId, [gslc\\_tsXKeyPadResult](#) \*psResult)  
*Callback function activated when a key has been pressed.*

### 9.14.1 Function Documentation

#### 9.14.1.1 `gslc_ElemXKeyPadBtnEvt_Alpha()`

```
void gslc_ElemXKeyPadBtnEvt_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    gslc_tsXKeyPadResult * psResult )
```

Callback function activated when a key has been pressed.

This callback is used to enable the KeyPad variant to handle any events associated with the key press and update any internal state.

- The callback is also used to determine whether any redraw actions need to be taken.

##### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>psResult</i>	The returned state vector (including redraw)

##### Returns

none

#### 9.14.1.2 `gslc_ElemXKeyPadCfgInit_Alpha()`

```
gslc_tsXKeyPadCfg_Alpha gslc_ElemXKeyPadCfgInit_Alpha ( )
```

Initialize the KeyPad config structure.

- This routine should be called to initialize the configuration data structure before calling any of the KeyPad config APIs

##### Returns

Initialized KeyPad config structure

#### 9.14.1.3 `gslc_ElemXKeyPadCreate_Alpha()`

```
gslc_tsElemRef* gslc_ElemXKeyPadCreate_Alpha (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pXData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg_Alpha * pConfig )
```

Create a KeyPad Element.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Ptr to config options

## Returns

Pointer to Element or NULL if failure

## 9.14.1.4 gslc\_ElemXKeyPadLabelGet\_Alpha()

```
void gslc_ElemXKeyPadLabelGet_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    uint8_t nStrMax,
    char * pStr )
```

Callback function to retrieve the label associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

## Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
in	<i>nStrMax</i>	Maximum length of return string (including NULL)
out	<i>pStr</i>	Buffer for the returned label

## Returns

none

## 9.14.1.5 gslc\_ElemXKeyPadReset\_Alpha()

```
void gslc_ElemXKeyPadReset_Alpha (
    void * pvConfig )
```

Callback function to reset internal state.

**Parameters**

in	<i>pvConfig</i>	Void ptr to the KeyPad config
----	-----------------	-------------------------------

**Returns**

none

**9.14.1.6 gslc\_ElemXKeyPadStyleGet\_Alpha()**

```
void gslc_ElemXKeyPadStyleGet_Alpha (
    void * pvKeyPad,
    uint8_t nId,
    bool * pbVisible,
    gslc_tsColor * pcolTxt,
    gslc_tsColor * pcolFrame,
    gslc_tsColor * pcolFill,
    gslc_tsColor * pcolGlow )
```

Callback function to retrieve the style associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

- This function is used to assign the color and visibility state of the keys at runtime.
- This function can also be used to change the appearance dynamically, according to internal state (eg. dimmed buttons).

**Parameters**

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>pbVisible</i>	The returned visibility state
out	<i>pcolTxt</i>	The returned text color
out	<i>pcolFrame</i>	The returned key's frame color
out	<i>pcolFill</i>	The returned key's fill color
out	<i>pcolGlow</i>	The returned key's fill color when highlighted

**Returns**

none

**9.14.1.7 gslc\_ElemXKeyPadTxtInit\_Alpha()**

```
void gslc_ElemXKeyPadTxtInit_Alpha (
    void * pvKeyPad )
```

Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).

- This is used to ensure any KeyPad variant state can be kept in sync with the text string.
- For example, if a numeric KeyPad is initiaized with a string that contains a minus sign, an internal negation flag might be set.

#### Parameters

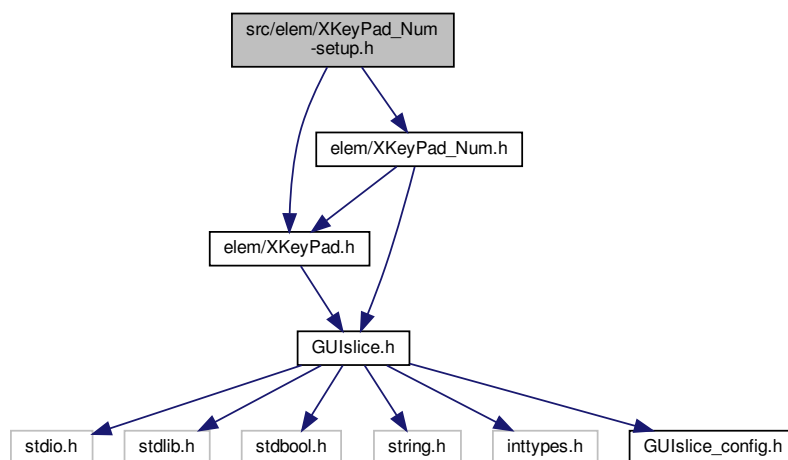
in	<i>pvKeyPad</i>	Void ptr to the KeyPad
----	-----------------	------------------------

#### Returns

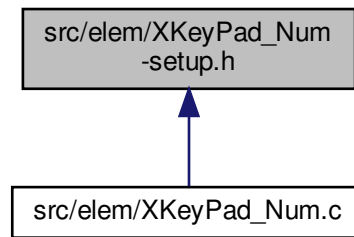
none

## 9.15 src/elem/XKeyPad\_Num-setup.h File Reference

```
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Num.h"
Include dependency graph for XKeyPad_Num-setup.h:
```



This graph shows which files directly or indirectly include this file:



## Macros

- `#define XKEYPAD_EXTEND_CHAR`
- `#define XKEYPAD_LABEL_MAX`
- `#define XKEYPAD_DISP_MAX`
- `#define XKEYPAD_KEY_W`
- `#define XKEYPAD_KEY_H`
- `#define XKEYPAD_SPACING_X`
- `#define XKEYPAD_SPACING_Y`
- `#define XKEYPAD_COL_DISABLE_TXT`
- `#define XKEYPAD_COL_DISABLE_FILL`
- `#define XKEYPAD_COL_DEF_TXT`
- `#define XKEYPAD_COL_DEF_FRAME`
- `#define XKEYPAD_COL_DEF_FILL`
- `#define XKEYPAD_COL_DEF_GLOW`
- `#define XKEYPAD_COL_BASIC_FILL`
- `#define XKEYPAD_COL_BASIC_GLOW`
- `#define XKEYPAD_COL_TEXT_TXT`
- `#define XKEYPAD_COL_TEXT_FILL`
- `#define XKEYPAD_COL_TEXT_GLOW`
- `#define XKEYPAD_COL_SPACE_FILL`
- `#define XKEYPAD_COL_SPACE_GLOW`
- `#define XKEYPAD_COL_ESC_FILL`
- `#define XKEYPAD_COL_ESC_GLOW`
- `#define XKEYPAD_COL_ENTER_FILL`
- `#define XKEYPAD_COL_ENTER_GLOW`
- `#define XKEYPAD_COL_SCROLL_L_FILL`
- `#define XKEYPAD_COL_SCROLL_L_GLOW`
- `#define XKEYPAD_COL_SCROLL_R_FILL`
- `#define XKEYPAD_COL_SCROLL_R_GLOW`
- `#define XKEYPAD_COL_DECIMAL_FILL`
- `#define XKEYPAD_COL_DECIMAL_GLOW`
- `#define XKEYPAD_COL_MINUS_FILL`
- `#define XKEYPAD_COL_MINUS_GLOW`

## Enumerations

- enum [gslc\\_teXKeyPadSel](#) {  
[E\\_XKEYPAD\\_SET\\_UPPER](#), [E\\_XKEYPAD\\_SET\\_LOWER](#), [E\\_XKEYPAD\\_SET\\_NUM](#), [E\\_XKEYPAD\\_SET\\_\\_MAX](#),  
[E\\_XKEYPAD\\_SET\\_NUM](#), [E\\_XKEYPAD\\_SET\\_\\_MAX](#) }
- enum { [KEYPAD\\_IDV\\_DECIMAL](#), [KEYPAD\\_IDV\\_MINUS](#) }

## Variables

- static const char \* [KEYPAD\\_LABEL\\_NEGATIVE](#)
- static const char \* [KEYPAD\\_LABEL\\_DECIMAL\\_PT](#)
- static const int8\_t [XKEYPAD\\_LAYOUT\\_DEFAULT](#)
- static [gslc\\_tsLabelSpecial](#) [KEYPAD\\_SPECIAL\\_LABEL](#) []
- static const char \* [KEYPAD\\_SET\\_LABEL](#) []
- static [gslc\\_tsKey](#) [KEYPAD\\_LAYOUT](#) []
- static [gslc\\_tsKey](#) \* [KEYPAD\\_LAYOUTS](#) [[E\\_XKEYPAD\\_SET\\_\\_MAX](#)]

## 9.15.1 Macro Definition Documentation

### 9.15.1.1 XKEYPAD\_COL\_BASIC\_FILL

```
#define XKEYPAD_COL_BASIC_FILL
```

### 9.15.1.2 XKEYPAD\_COL\_BASIC\_GLOW

```
#define XKEYPAD_COL_BASIC_GLOW
```

### 9.15.1.3 XKEYPAD\_COL\_DECIMAL\_FILL

```
#define XKEYPAD_COL_DECIMAL_FILL
```

### 9.15.1.4 XKEYPAD\_COL\_DECIMAL\_GLOW

```
#define XKEYPAD_COL_DECIMAL_GLOW
```

#### 9.15.1.5 XKEYPAD\_COL\_DEF\_FILL

```
#define XKEYPAD_COL_DEF_FILL
```

#### 9.15.1.6 XKEYPAD\_COL\_DEF\_FRAME

```
#define XKEYPAD_COL_DEF_FRAME
```

#### 9.15.1.7 XKEYPAD\_COL\_DEF\_GLOW

```
#define XKEYPAD_COL_DEF_GLOW
```

#### 9.15.1.8 XKEYPAD\_COL\_DEF\_TXT

```
#define XKEYPAD_COL_DEF_TXT
```

#### 9.15.1.9 XKEYPAD\_COL\_DISABLE\_FILL

```
#define XKEYPAD_COL_DISABLE_FILL
```

#### 9.15.1.10 XKEYPAD\_COL\_DISABLE\_TXT

```
#define XKEYPAD_COL_DISABLE_TXT
```

#### 9.15.1.11 XKEYPAD\_COL\_ENTER\_FILL

```
#define XKEYPAD_COL_ENTER_FILL
```

#### 9.15.1.12 XKEYPAD\_COL\_ENTER\_GLOW

```
#define XKEYPAD_COL_ENTER_GLOW
```

**9.15.1.13 XKEYPAD\_COL\_ESC\_FILL**

```
#define XKEYPAD_COL_ESC_FILL
```

**9.15.1.14 XKEYPAD\_COL\_ESC\_GLOW**

```
#define XKEYPAD_COL_ESC_GLOW
```

**9.15.1.15 XKEYPAD\_COL\_MINUS\_FILL**

```
#define XKEYPAD_COL_MINUS_FILL
```

**9.15.1.16 XKEYPAD\_COL\_MINUS\_GLOW**

```
#define XKEYPAD_COL_MINUS_GLOW
```

**9.15.1.17 XKEYPAD\_COL\_SCROLL\_L\_FILL**

```
#define XKEYPAD_COL_SCROLL_L_FILL
```

**9.15.1.18 XKEYPAD\_COL\_SCROLL\_L\_GLOW**

```
#define XKEYPAD_COL_SCROLL_L_GLOW
```

**9.15.1.19 XKEYPAD\_COL\_SCROLL\_R\_FILL**

```
#define XKEYPAD_COL_SCROLL_R_FILL
```

**9.15.1.20 XKEYPAD\_COL\_SCROLL\_R\_GLOW**

```
#define XKEYPAD_COL_SCROLL_R_GLOW
```

**9.15.1.21 XKEYPAD\_COL\_SPACE\_FILL**

```
#define XKEYPAD_COL_SPACE_FILL
```

**9.15.1.22 XKEYPAD\_COL\_SPACE\_GLOW**

```
#define XKEYPAD_COL_SPACE_GLOW
```

**9.15.1.23 XKEYPAD\_COL\_TEXT\_FILL**

```
#define XKEYPAD_COL_TEXT_FILL
```

**9.15.1.24 XKEYPAD\_COL\_TEXT\_GLOW**

```
#define XKEYPAD_COL_TEXT_GLOW
```

**9.15.1.25 XKEYPAD\_COL\_TEXT\_TXT**

```
#define XKEYPAD_COL_TEXT_TXT
```

**9.15.1.26 XKEYPAD\_DISP\_MAX**

```
#define XKEYPAD_DISP_MAX
```

**9.15.1.27 XKEYPAD\_EXTEND\_CHAR**

```
#define XKEYPAD_EXTEND_CHAR
```

**9.15.1.28 XKEYPAD\_KEY\_H**

```
#define XKEYPAD_KEY_H
```



#### 9.15.1.29 XKEYPAD\_KEY\_W

```
#define XKEYPAD_KEY_W
```

#### 9.15.1.30 XKEYPAD\_LABEL\_MAX

```
#define XKEYPAD_LABEL_MAX
```

#### 9.15.1.31 XKEYPAD\_SPACING\_X

```
#define XKEYPAD_SPACING_X
```

#### 9.15.1.32 XKEYPAD\_SPACING\_Y

```
#define XKEYPAD_SPACING_Y
```

### 9.15.2 Enumeration Type Documentation

#### 9.15.2.1 anonymous enum

anonymous enum

##### Enumerator

KEYPAD_IDV_DECIMAL	
KEYPAD_IDV_MINUS	

#### 9.15.2.2 gslc\_teXKeyPadSel

enum `gslc_teXKeyPadSel`

##### Enumerator

E_XKEYPAD_SET_UPPER	
---------------------	--

## Enumerator

E_XKEYPAD_SET_LOWER	
E_XKEYPAD_SET_NUM	
E_XKEYPAD_SET__MAX	
E_XKEYPAD_SET_NUM	
E_XKEYPAD_SET__MAX	

### 9.15.3 Variable Documentation

#### 9.15.3.1 KEYPAD\_LABEL\_DECIMAL\_PT

```
const char* KEYPAD_LABEL_DECIMAL_PT [static]
```

#### 9.15.3.2 KEYPAD\_LABEL\_NEGATIVE

```
const char* KEYPAD_LABEL_NEGATIVE [static]
```

#### 9.15.3.3 KEYPAD\_LAYOUT

```
gslc_tsKey KEYPAD_LAYOUT[] [static]
```

#### 9.15.3.4 KEYPAD\_LAYOUTS

```
gslc_tsKey* KEYPAD_LAYOUTS[E_XKEYPAD_SET__MAX] [static]
```

#### 9.15.3.5 KEYPAD\_SET\_LABEL

```
const char* KEYPAD_SET_LABEL[] [static]
```

## 9.15.3.6 KEYPAD\_SPECIAL\_LABEL

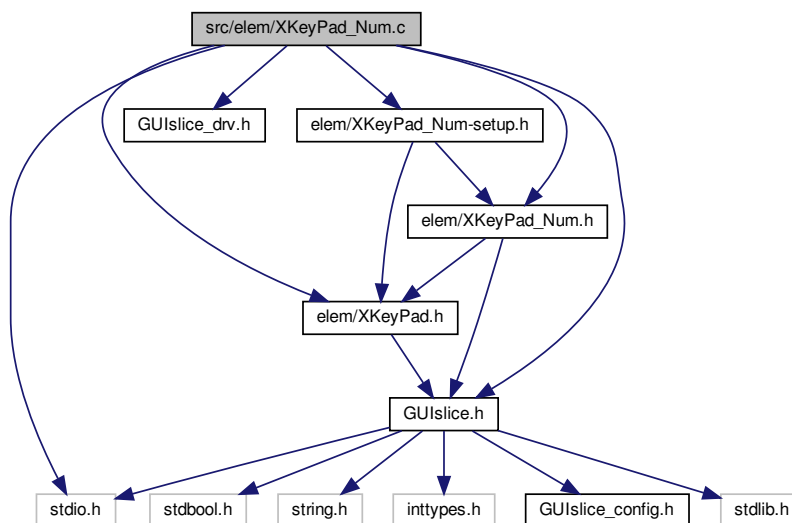
```
gslc_tsLabelSpecial KEYPAD_SPECIAL_LABEL[] [static]
```

## 9.15.3.7 XKEYPAD\_LAYOUT\_DEFAULT

```
const int8_t XKEYPAD_LAYOUT_DEFAULT [static]
```

## 9.16 src/elem/XKeyPad\_Num.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Num.h"
#include <stdio.h>
#include "elem/XKeyPad_Num-setup.h"
Include dependency graph for XKeyPad_Num.c:
```



## Functions

- void [gslc\\_XKeyPadValSetSign\\_Num](#) ([gslc\\_tsXKeyPad](#) \*pXKeyPad, bool bPositive)
- void [gslc\\_ElemXKeyPadReset\\_Num](#) (void \*pvConfig)  
*Callback function to reset internal state.*
- void [gslc\\_ElemXKeyPadTxtInit\\_Num](#) (void \*pvKeyPad)  
*Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).*
- void [gslc\\_ElemXKeyPadLabelGet\\_Num](#) (void \*pvKeyPad, uint8\_t nId, uint8\_t nStrMax, char \*pStr)  
*Callback function to retrieve the label associated with a KeyPad button.*

- void `gslc_ElemXKeyPadStyleGet_Num` (void \*pvKeyPad, uint8\_t nId, bool \*pbVisible, `gslc_tsColor` \*pcolTxt, `gslc_tsColor` \*pcolFrame, `gslc_tsColor` \*pcolFill, `gslc_tsColor` \*pcolGlow)  
*Callback function to retrieve the style associated with a KeyPad button.*
- void `gslc_ElemXKeyPadBtnEvt_Num` (void \*pvKeyPad, uint8\_t nId, `gslc_tsXKeyPadResult` \*psResult)  
*Callback function activated when a key has been pressed.*
- `gslc_tsXKeyPadCfg_Num` `gslc_ElemXKeyPadCfgInit_Num` ()  
*Initialize the KeyPad config structure.*
- `gslc_tsElemRef` \* `gslc_ElemXKeyPadCreate_Num` (`gslc_tsGui` \*pGui, int16\_t nElemId, int16\_t nPage, `gslc_tsXKeyPad` \*pXData, int16\_t nX0, int16\_t nY0, int8\_t nFontId, `gslc_tsXKeyPadCfg_Num` \*pConfig)  
*Create a KeyPad Element.*
- void `gslc_ElemXKeyPadCfgSetFloatEn_Num` (`gslc_tsXKeyPadCfg_Num` \*pConfig, bool bEn)  
*Update the KeyPad configuration to enable floating point numbers.*
- void `gslc_ElemXKeyPadCfgSetSignEn_Num` (`gslc_tsXKeyPadCfg_Num` \*pConfig, bool bEn)  
*Update the KeyPad configuration to enable negative numbers.*

## Variables

- const char `GSLC_PMEM ERRSTR_NULL` []
- const char `GSLC_PMEM ERRSTR_PXD_NULL` []

## 9.16.1 Function Documentation

### 9.16.1.1 `gslc_ElemXKeyPadBtnEvt_Num()`

```
void gslc_ElemXKeyPadBtnEvt_Num (
    void * pvKeyPad,
    uint8_t nId,
    gslc_tsXKeyPadResult * psResult )
```

Callback function activated when a key has been pressed.

This callback is used to enable the KeyPad variant to handle any events associated with the key press and update any internal state.

- The callback is also used to determine whether any redraw actions need to be taken.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>psResult</i>	The returned state vector (including redraw)

#### Returns

none

### 9.16.1.2 gslc\_ElemXKeyPadCfgInit\_Num()

```
gslc_tsXKeyPadCfg_Num gslc_ElemXKeyPadCfgInit_Num ( )
```

Initialize the KeyPad config structure.

- This routine should be called to initialize the configuration data structure before calling any of the KeyPad config APIs

#### Returns

Initialized KeyPad config structure

### 9.16.1.3 gslc\_ElemXKeyPadCfgSetFloatEn\_Num()

```
void gslc_ElemXKeyPadCfgSetFloatEn_Num (
    gslc_tsXKeyPadCfg_Num * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable floating point numbers.

- Effectively disables/enables the decimal point button & handling

#### Parameters

in	<i>pConfig</i>	Pointer to the XKeyPad variant config structure
in	<i>bEn</i>	Enable flag (true if floating point enabled)

#### Returns

none

### 9.16.1.4 gslc\_ElemXKeyPadCfgSetSignEn\_Num()

```
void gslc_ElemXKeyPadCfgSetSignEn_Num (
    gslc_tsXKeyPadCfg_Num * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable negative numbers.

- Effectively disables/enables the sign button & handling

## Parameters

in	<i>pConfig</i>	Pointer to the XKeyPad variant config structure
in	<i>bEn</i>	Enable flag (true if negative numbers enabled)

## Returns

none

## 9.16.1.5 gslc\_ElemXKeyPadCreate\_Num()

```
gslc_tsElemRef* gslc_ElemXKeyPadCreate_Num (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pXData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg_Num * pConfig )
```

Create a KeyPad Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Ptr to config options

## Returns

Pointer to Element or NULL if failure

## 9.16.1.6 gslc\_ElemXKeyPadLabelGet\_Num()

```
void gslc_ElemXKeyPadLabelGet_Num (
    void * pvKeyPad,
    uint8_t nId,
```

```
uint8_t nStrMax,
char * pStr )
```

Callback function to retrieve the label associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
in	<i>nStrMax</i>	Maximum length of return string (including NULL)
out	<i>pStr</i>	Buffer for the returned label

#### Returns

none

#### 9.16.1.7 gslc\_ElemXKeyPadReset\_Num()

```
void gslc_ElemXKeyPadReset_Num (
    void * pvConfig )
```

Callback function to reset internal state.

#### Parameters

in	<i>pvConfig</i>	Void ptr to the KeyPad config
----	-----------------	-------------------------------

#### Returns

none

#### 9.16.1.8 gslc\_ElemXKeyPadStyleGet\_Num()

```
void gslc_ElemXKeyPadStyleGet_Num (
    void * pvKeyPad,
    uint8_t nId,
    bool * pbVisible,
    gslc_tsColor * pcolTxt,
    gslc_tsColor * pcolFrame,
    gslc_tsColor * pcolFill,
    gslc_tsColor * pcolGlow )
```

Callback function to retrieve the style associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

- This function is used to assign the color and visibility state of the keys at runtime.
- This function can also be used to change the appearance dynamically, according to internal state (eg. dimmed buttons).

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>pbVisible</i>	The returned visibility state
out	<i>pcolTxt</i>	The returned text color
out	<i>pcolFrame</i>	The returned key's frame color
out	<i>pcolFill</i>	The returned key's fill color
out	<i>pcolGlow</i>	The returned key's fill color when highlighted

#### Returns

none

#### 9.16.1.9 gslc\_ElemXKeyPadTxtInit\_Num()

```
void gslc_ElemXKeyPadTxtInit_Num (
    void * pvKeyPad )
```

Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).

- This is used to ensure any KeyPad variant state can be kept in sync with the text string.
- For example, if a numeric KeyPad is initiaized with a string that contains a minus sign, an internal negation flag might be set.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
----	-----------------	------------------------

#### Returns

none

#### 9.16.1.10 gslc\_XKeyPadValSetSign\_Num()

```
void gslc_XKeyPadValSetSign_Num (
    gslc_tsXKeyPad * pXKeyPad,
    bool bPositive )
```



## 9.16.2 Variable Documentation

### 9.16.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

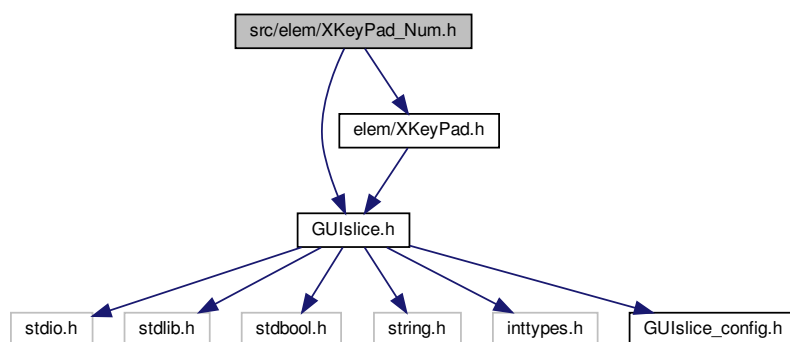
### 9.16.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

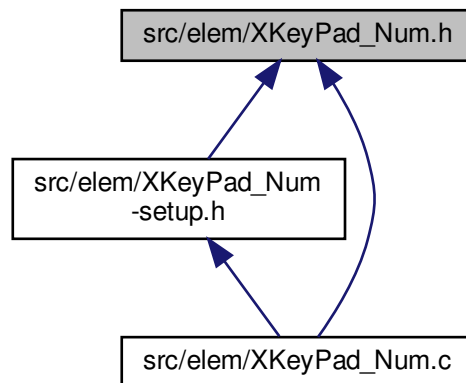
## 9.17 src/elem/XKeyPad\_Num.h File Reference

```
#include "GUIslice.h"  
#include "elem/XKeyPad.h"
```

Include dependency graph for XKeyPad\_Num.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXKeyPadCfg\\_Num](#)

## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXKeyPadCreate\\_Num](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nElemId, [int16\\_t](#) nPage, [gslc\\_tsXKeyPad](#) \*pXData, [int16\\_t](#) nX0, [int16\\_t](#) nY0, [int8\\_t](#) nFontId, [gslc\\_tsXKeyPadCfg\\_Num](#) \*pConfig)  
*Create a KeyPad Element.*
- [gslc\\_tsXKeyPadCfg\\_Num gslc\\_ElemXKeyPadCfgInit\\_Num](#) ()  
*Initialize the KeyPad config structure.*
- void [gslc\\_ElemXKeyPadReset\\_Num](#) (void \*pvConfig)  
*Callback function to reset internal state.*
- void [gslc\\_ElemXKeyPadTxtInit\\_Num](#) (void \*pvKeyPad)  
*Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).*
- void [gslc\\_ElemXKeyPadLabelGet\\_Num](#) (void \*pvKeyPad, [uint8\\_t](#) nId, [uint8\\_t](#) nStrMax, char \*pStr)  
*Callback function to retrieve the label associated with a KeyPad button.*
- void [gslc\\_ElemXKeyPadStyleGet\\_Num](#) (void \*pvKeyPad, [uint8\\_t](#) nId, bool \*pbVisible, [gslc\\_tsColor](#) \*pcolTxt, [gslc\\_tsColor](#) \*pcolFrame, [gslc\\_tsColor](#) \*pcolFill, [gslc\\_tsColor](#) \*pcolGlow)  
*Callback function to retrieve the style associated with a KeyPad button.*
- void [gslc\\_ElemXKeyPadBtnEvt\\_Num](#) (void \*pvKeyPad, [uint8\\_t](#) nId, [gslc\\_tsXKeyPadResult](#) \*psResult)  
*Callback function activated when a key has been pressed.*
- void [gslc\\_ElemXKeyPadCfgSetFloatEn\\_Num](#) ([gslc\\_tsXKeyPadCfg\\_Num](#) \*pConfig, bool bEn)  
*Update the KeyPad configuration to enable floating point numbers.*
- void [gslc\\_ElemXKeyPadCfgSetSignEn\\_Num](#) ([gslc\\_tsXKeyPadCfg\\_Num](#) \*pConfig, bool bEn)  
*Update the KeyPad configuration to enable negative numbers.*

### 9.17.1 Function Documentation

#### 9.17.1.1 gslc\_ElemXKeyPadBtnEvt\_Num()

```
void gslc_ElemXKeyPadBtnEvt_Num (
    void * pvKeyPad,
    uint8_t nId,
    gslc_tsXKeyPadResult * psResult )
```

Callback function activated when a key has been pressed.

This callback is used to enable the KeyPad variant to handle any events associated with the key press and update any internal state.

- The callback is also used to determine whether any redraw actions need to be taken.

##### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>psResult</i>	The returned state vector (including redraw)

##### Returns

none

#### 9.17.1.2 gslc\_ElemXKeyPadCfgInit\_Num()

```
gslc_tsXKeyPadCfg_Num gslc_ElemXKeyPadCfgInit_Num ( )
```

Initialize the KeyPad config structure.

- This routine should be called to initialize the configuration data structure before calling any of the KeyPad config APIs

##### Returns

Initialized KeyPad config structure

#### 9.17.1.3 gslc\_ElemXKeyPadCfgSetFloatEn\_Num()

```
void gslc_ElemXKeyPadCfgSetFloatEn_Num (
    gslc_tsXKeyPadCfg_Num * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable floating point numbers.

- Effectively disables/enables the decimal point button & handling

**Parameters**

in	<i>pConfig</i>	Pointer to the XKeyPad variant config structure
in	<i>bEn</i>	Enable flag (true if floating point enabled)

**Returns**

none

**9.17.1.4 gslc\_ElemXKeyPadCfgSetSignEn\_Num()**

```
void gslc_ElemXKeyPadCfgSetSignEn_Num (
    gslc_tsXKeyPadCfg_Num * pConfig,
    bool bEn )
```

Update the KeyPad configuration to enable negative numbers.

- Effectively disables/enables the sign button & handling

**Parameters**

in	<i>pConfig</i>	Pointer to the XKeyPad variant config structure
in	<i>bEn</i>	Enable flag (true if negative numbers enabled)

**Returns**

none

**9.17.1.5 gslc\_ElemXKeyPadCreate\_Num()**

```
gslc_tsElemRef* gslc_ElemXKeyPadCreate_Num (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXKeyPad * pData,
    int16_t nX0,
    int16_t nY0,
    int8_t nFontId,
    gslc_tsXKeyPadCfg_Num * pConfig )
```

Create a KeyPad Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>nX0</i>	X KeyPad Starting Coordinate
in	<i>nY0</i>	Y KeyPad Starting Coordinate
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>pConfig</i>	Ptr to config options

## Returns

Pointer to Element or NULL if failure

## 9.17.1.6 gslc\_ElemXKeyPadLabelGet\_Num()

```
void gslc_ElemXKeyPadLabelGet_Num (
    void * pvKeyPad,
    uint8_t nId,
    uint8_t nStrMax,
    char * pStr )
```

Callback function to retrieve the label associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

## Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
in	<i>nStrMax</i>	Maximum length of return string (including NULL)
out	<i>pStr</i>	Buffer for the returned label

## Returns

none

## 9.17.1.7 gslc\_ElemXKeyPadReset\_Num()

```
void gslc_ElemXKeyPadReset_Num (
    void * pvConfig )
```

Callback function to reset internal state.

**Parameters**

in	<i>pvConfig</i>	Void ptr to the KeyPad config
----	-----------------	-------------------------------

**Returns**

none

**9.17.1.8 gslc\_ElemXKeyPadStyleGet\_Num()**

```
void gslc_ElemXKeyPadStyleGet_Num (
    void * pvKeyPad,
    uint8_t nId,
    bool * pbVisible,
    gslc_tsColor * pcolTxt,
    gslc_tsColor * pcolFrame,
    gslc_tsColor * pcolFill,
    gslc_tsColor * pcolGlow )
```

Callback function to retrieve the style associated with a KeyPad button.

This is called during the drawing of the KeyPad layout.

- This function is used to assign the color and visibility state of the keys at runtime.
- This function can also be used to change the appearance dynamically, according to internal state (eg. dimmed buttons).

**Parameters**

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
in	<i>nId</i>	KeyPad key ID
out	<i>pbVisible</i>	The returned visibility state
out	<i>pcolTxt</i>	The returned text color
out	<i>pcolFrame</i>	The returned key's frame color
out	<i>pcolFill</i>	The returned key's fill color
out	<i>pcolGlow</i>	The returned key's fill color when highlighted

**Returns**

none

**9.17.1.9 gslc\_ElemXKeyPadTxtInit\_Num()**

```
void gslc_ElemXKeyPadTxtInit_Num (
    void * pvKeyPad )
```

Callback function to update internal state whenever the text field is manually set via [gslc\\_ElemXKeyPadValSet\(\)](#).

- This is used to ensure any KeyPad variant state can be kept in sync with the text string.
- For example, if a numeric KeyPad is initiaized with a string that contains a minus sign, an internal negation flag might be set.

#### Parameters

in	<i>pvKeyPad</i>	Void ptr to the KeyPad
----	-----------------	------------------------

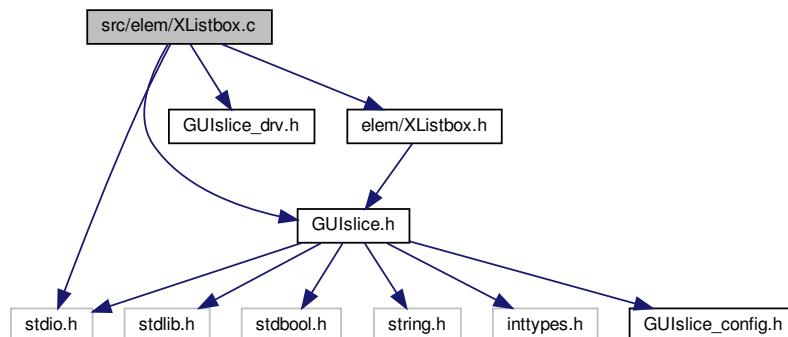
#### Returns

none

## 9.18 src/elem/XListbox.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XListbox.h"
#include <stdio.h>
```

Include dependency graph for XListbox.c:



#### Macros

- `#define` [XLISTBOX\\_MAX\\_STR](#)

#### Functions

- `char *` [gslc\\_ElemXListboxGetItemAddr](#) ([gslc\\_tsXListbox](#) \*pListbox, `int16_t` nItemCurSel)
- `bool` [gslc\\_ElemXListboxRecalcSize](#) ([gslc\\_tsXListbox](#) \*pListbox, [gslc\\_tsRect](#) rElem)
- `void` [gslc\\_ElemXListboxSetSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, `int8_t` nRows, `int8_t` nCols)  
*Configure the number of rows & columns to display in the listbox.*

- void [gslc\\_ElemXListboxSetMargin](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nMarginW, int8\_t nMarginH)  
*Configure the margin inside the listbox.*
- void [gslc\\_ElemXListboxItemsSetSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemW, int16\_t nItemH)  
*Configure the size of the listbox items.*
- void [gslc\\_ElemXListboxItemsSetGap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nGap, [gslc\\_tsColor](#) colGap)  
*Configure the gap between listbox items.*
- void [gslc\\_ElemXListboxReset](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Empty the listbox of all items.*
- bool [gslc\\_ElemXListboxAddItem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStrItem)  
*Add an item to the listbox.*
- bool [gslc\\_ElemXListboxInsertItemAt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nInsertPos, const char \*pStrItem)  
*Insert an item in the listbox at a specific position.*
- bool [gslc\\_ElemXListboxDeleteItemAt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nDeletePos)  
*Insert an item in the listbox at a specific position.*
- bool [gslc\\_ElemXListboxGetItem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemCurSel, char \*pStrItem, uint8\_t nStrItemLen)  
*Get the indexed listbox item.*
- int16\_t [gslc\\_ElemXListboxGetItemCnt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the number of items in the listbox.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXListboxCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXListbox](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nFontId, uint8\_t \*pBufItems, uint16\_t nBufItemsMax, int16\_t nItemDefault)  
*Create a Listbox Element.*
- bool [gslc\\_ElemXListboxDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Listbox element on the screen.*
- bool [gslc\\_ElemXListboxTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)  
*Handle touch events to Listbox element.*
- int16\_t [gslc\\_ElemXListboxGetSel](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a Listbox element's current selection.*
- bool [gslc\\_ElemXListboxSetSel](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemCurSel)  
*Set a Listbox element's current selection.*
- bool [gslc\\_ElemXListboxSetScrollPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nScrollPos)  
*Set the Listbox scroll position.*
- void [gslc\\_ElemXListboxSetSelFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_XLISTBOX\\_SEL](#) funcCb)  
*Assign the selection callback function for a Listbox.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.18.1 Macro Definition Documentation



### 9.18.1.1 XLISTBOX\_MAX\_STR

```
#define XLISTBOX_MAX_STR
```

## 9.18.2 Function Documentation

### 9.18.2.1 gslc\_ElemXListboxAddItem()

```
bool gslc_ElemXListboxAddItem (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    const char * pStrItem )
```

Add an item to the listbox.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>pStrItem</i>	String to use when creating the listbox item

#### Returns

true if OK, false if fail (eg. insufficient buffer storage)

### 9.18.2.2 gslc\_ElemXListboxCreate()

```
gslc_tsElemRef* gslc_ElemXListboxCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXListbox * pXData,
    gslc_tsRect rElem,
    int16_t nFontId,
    uint8_t * pBufItems,
    uint16_t nBufItemsMax,
    int16_t nSelDefault )
```

Create a Listbox Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to

**Parameters**

in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nFontId</i>	Font ID for item display
in	<i>pBufItems</i>	Pointer to buffer that will contain list of items
in	<i>nBufItemsMax</i>	Max size of buffer for list of items (pBufItems)
in	<i>nSelDefault</i>	Default item to select

**Returns**

Pointer to Element reference or NULL if failure

**9.18.2.3 gslc\_ElemXListboxDeleteItemAt()**

```
bool gslc_ElemXListboxDeleteItemAt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nDeletePos )
```

Insert an item in the listbox at a specific position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nDeletePos</i>	Position to delete

**Returns**

true if OK, false if fail

**9.18.2.4 gslc\_ElemXListboxDraw()**

```
bool gslc_ElemXListboxDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Listbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.18.2.5 `gslc_ElemXListboxGetItem()`

```
bool gslc_ElemXListboxGetItem (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemCurSel,
    char * pStrItem,
    uint8_t nStrItemLen )
```

Get the indexed listbox item.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nItemCurSel</i>	Item index to fetch
out	<i>pStrItem</i>	Ptr to the string buffer to receive the item
in	<i>nStrItemLen</i>	Maximum buffer length of pStrItem

## Returns

true if success, false if fail (eg. can't locate item)

9.18.2.6 `gslc_ElemXListboxGetItemAddr()`

```
char* gslc_ElemXListboxGetItemAddr (
    gslc_tsXListbox * pListbox,
    int16_t nItemCurSel )
```

9.18.2.7 `gslc_ElemXListboxGetItemCnt()`

```
int16_t gslc_ElemXListboxGetItemCnt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the number of items in the listbox.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update

**Returns**

Number of items

**9.18.2.8 gslc\_ElemXListboxGetSel()**

```
int16_t gslc_ElemXListboxGetSel (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Listbox element's current selection.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Current Listbox selection (or -1 if none)

**9.18.2.9 gslc\_ElemXListboxInsertItemAt()**

```
bool gslc_ElemXListboxInsertItemAt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nInsertPos,
    const char * pStrItem )
```

Insert an item in the listbox at a specific position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nInsertPos</i>	Insertion position
in	<i>pStrItem</i>	String to use when creating the listbox item

**Returns**

true if OK, false if fail (eg. insufficient buffer storage)

**9.18.2.10 gslc\_ElemXListboxItemsSetGap()**

```
void gslc_ElemXListboxItemsSetGap (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nGap,
    gslc_tsColor colGap )
```

Configure the gap between listbox items.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nGap</i>	Set the gap between listbox items (0 for none)
in	<i>colGap</i>	Set the color of the gap between listbox items

**Returns**

none

**9.18.2.11 gslc\_ElemXListboxItemsSetSize()**

```
void gslc_ElemXListboxItemsSetSize (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemW,
    int16_t nItemH )
```

Configure the size of the listbox items.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nItemW</i>	Set the width of a listbox item (or -1 to auto-size)
in	<i>nItemH</i>	Set the height of a listbox item

**Returns**

none

#### 9.18.2.12 gslc\_ElemXListboxRecalcSize()

```
bool gslc_ElemXListboxRecalcSize (
    gslc_tsXListbox * pListbox,
    gslc_tsRect rElem )
```

#### 9.18.2.13 gslc\_ElemXListboxReset()

```
void gslc_ElemXListboxReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Empty the listbox of all items.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update

##### Returns

none

#### 9.18.2.14 gslc\_ElemXListboxSetMargin()

```
void gslc_ElemXListboxSetMargin (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nMarginW,
    int8_t nMarginH )
```

Configure the margin inside the listbox.

- Defines the region bewteen the element rect and the inner listbox items

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nMarginW</i>	Set the margin (horizontal) inside the listbox (0 for none)
in	<i>nMarginH</i>	Set the margin (horizontal) inside the listbox (0 for none)

**Returns**

none

**9.18.2.15 gslc\_ElemXListboxSetScrollPos()**

```
bool gslc_ElemXListboxSetScrollPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nScrollPos )
```

Set the Listbox scroll position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	Scroll the listbox so that the nScrollPos item is at the top (0 default)

**Returns**

true if success, false if fail

**9.18.2.16 gslc\_ElemXListboxSetSel()**

```
bool gslc_ElemXListboxSetSel (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemCurSel )
```

Set a Listbox element's current selection.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nItemCurSel</i>	Listbox item to select (or -1 for none)

**Returns**

true if success, false if fail

### 9.18.2.17 gslc\_ElemXListboxSetSelFunc()

```
void gslc_ElemXListboxSetSelFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XLISTBOX_SEL funcCb )
```

Assign the selection callback function for a Listbox.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to selection routine (or NULL for none)

#### Returns

none

### 9.18.2.18 gslc\_ElemXListboxSetSize()

```
void gslc_ElemXListboxSetSize (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nRows,
    int8_t nCols )
```

Configure the number of rows & columns to display in the listbox.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nRows</i>	Number of rows ( $\geq 1$ , or XLISTBOX_SIZE_AUTO to base on content)
in	<i>nCols</i>	Number of columns ( $\geq 1$ )

#### Returns

none

### 9.18.2.19 gslc\_ElemXListboxTouch()

```
bool gslc_ElemXListboxTouch (
    void * pvGui,
```



```
void * pvElemRef,
gslc_teTouch eTouch,
int16_t nRelX,
int16_t nRelY )
```

Handle touch events to Listbox element.

- Called from `gslc_ElemSendEventTouch()`

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

### 9.18.3 Variable Documentation

#### 9.18.3.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

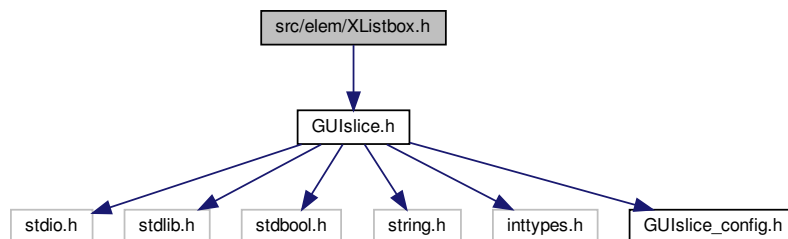
#### 9.18.3.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

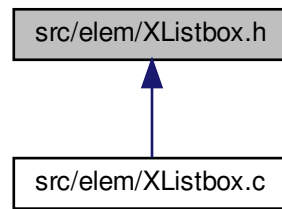
## 9.19 src/elem/XListbox.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XListbox.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXListbox](#)  
*Extended data for Listbox element.*

## Macros

- #define [GSLC\\_TYPEX\\_LISTBOX](#)
- #define [XLISTBOX\\_SEL\\_NONE](#)
- #define [XLISTBOX\\_SIZE\\_AUTO](#)
- #define [XLISTBOX\\_BUF\\_OH\\_R](#)

## Typedefs

- typedef bool(\* [GSLC\\_CB\\_XLISTBOX\\_SEL](#)) (void \*pvGui, void \*pvElem, int16\_t nSel)  
*Callback function for Listbox feedback.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXListboxCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXListbox](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nFontId, uint8\_t \*pBufItems, uint16\_t nBufItemsMax, int16\_t nSelDefault)  
*Create a Listbox Element.*
- void [gslc\\_ElemXListboxSetSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nRows, int8\_t nCols)  
*Configure the number of rows & columns to display in the listbox.*
- void [gslc\\_ElemXListboxSetMargin](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nMarginW, int8\_t nMarginH)  
*Configure the margin inside the listbox.*
- void [gslc\\_ElemXListboxItemsSetSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemW, int16\_t nItemH)  
*Configure the size of the listbox items.*
- void [gslc\\_ElemXListboxItemsSetGap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nGap, [gslc\\_tsColor](#) colGap)  
*Configure the gap between listbox items.*

- void [gslc\\_ElemXListboxReset](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Empty the listbox of all items.*
- bool [gslc\\_ElemXListboxAddItem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStrItem)  
*Add an item to the listbox.*
- bool [gslc\\_ElemXListboxInsertItemAt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nInsertPos, const char \*pStrItem)  
*Insert an item in the listbox at a specific position.*
- bool [gslc\\_ElemXListboxDeleteItemAt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nDeletePos)  
*Insert an item in the listbox at a specific position.*
- bool [gslc\\_ElemXListboxGetItem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemCurSel, char \*pStrItem, uint8\_t nStrItemLen)  
*Get the indexed listbox item.*
- int16\_t [gslc\\_ElemXListboxGetItemCnt](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the number of items in the listbox.*
- bool [gslc\\_ElemXListboxDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Listbox element on the screen.*
- bool [gslc\\_ElemXListboxTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)  
*Handle touch events to Listbox element.*
- int16\_t [gslc\\_ElemXListboxGetSel](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a Listbox element's current selection.*
- bool [gslc\\_ElemXListboxSetSel](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nItemCurSel)  
*Set a Listbox element's current selection.*
- bool [gslc\\_ElemXListboxSetScrollPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nScrollPos)  
*Set the Listbox scroll position.*
- void [gslc\\_ElemXListboxSetSelFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_XLISTBOX\\_SEL](#) funcCb)  
*Assign the selection callback function for a Listbox.*

## 9.19.1 Macro Definition Documentation

### 9.19.1.1 GSLC\_TYPEX\_LISTBOX

```
#define GSLC_TYPEX_LISTBOX
```

### 9.19.1.2 XLISTBOX\_BUF\_OH\_R

```
#define XLISTBOX_BUF_OH_R
```

### 9.19.1.3 XLISTBOX\_SEL\_NONE

```
#define XLISTBOX_SEL_NONE
```

#### 9.19.1.4 XLISTBOX\_SIZE\_AUTO

```
#define XLISTBOX_SIZE_AUTO
```

### 9.19.2 Typedef Documentation

#### 9.19.2.1 GSLC\_CB\_XLISTBOX\_SEL

```
typedef bool(* GSLC_CB_XLISTBOX_SEL) (void *pvGui, void *pvElem, int16_t nSel)
```

Callback function for Listbox feedback.

### 9.19.3 Function Documentation

#### 9.19.3.1 gslc\_ElemXListboxAddItem()

```
bool gslc_ElemXListboxAddItem (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    const char * pStrItem )
```

Add an item to the listbox.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>pStrItem</i>	String to use when creating the listbox item

##### Returns

true if OK, false if fail (eg. insufficient buffer storage)

#### 9.19.3.2 gslc\_ElemXListboxCreate()

```
gslc_tsElemRef* gslc_ElemXListboxCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
```

```

gslc_tsXListbox * pXData,
gslc_tsRect rElem,
int16_t nFontId,
uint8_t * pBufItems,
uint16_t nBufItemsMax,
int16_t nSelDefault )

```

Create a Listbox Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nFontId</i>	Font ID for item display
in	<i>pBufItems</i>	Pointer to buffer that will contain list of items
in	<i>nBufItemsMax</i>	Max size of buffer for list of items (pBufItems)
in	<i>nSelDefault</i>	Default item to select

#### Returns

Pointer to Element reference or NULL if failure

#### 9.19.3.3 gslc\_ElemXListboxDeleteItemAt()

```

bool gslc_ElemXListboxDeleteItemAt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nDeletePos )

```

Insert an item in the listbox at a specific position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nDeletePos</i>	Position to delete

#### Returns

true if OK, false if fail

#### 9.19.3.4 gslc\_ElemXListboxDraw()

```
bool gslc_ElemXListboxDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Listbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

##### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

##### Returns

true if success, false otherwise

#### 9.19.3.5 gslc\_ElemXListboxGetItem()

```
bool gslc_ElemXListboxGetItem (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemCurSel,
    char * pStrItem,
    uint8_t nStrItemLen )
```

Get the indexed listbox item.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nItemCurSel</i>	Item index to fetch
out	<i>pStrItem</i>	Ptr to the string buffer to receive the item
in	<i>nStrItemLen</i>	Maximum buffer length of <i>pStrItem</i>

##### Returns

true if success, false if fail (eg. can't locate item)

### 9.19.3.6 gslc\_ElemXListboxGetItemCnt()

```
int16_t gslc_ElemXListboxGetItemCnt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get the number of items in the listbox.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update

#### Returns

Number of items

### 9.19.3.7 gslc\_ElemXListboxGetSel()

```
int16_t gslc_ElemXListboxGetSel (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Listbox element's current selection.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

Current Listbox selection (or -1 if none)

### 9.19.3.8 gslc\_ElemXListboxInsertItemAt()

```
bool gslc_ElemXListboxInsertItemAt (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nInsertPos,
    const char * pStrItem )
```

Insert an item in the listbox at a specific position.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nInsertPos</i>	Insertion position
in	<i>pStrItem</i>	String to use when creating the listbox item

## Returns

true if OK, false if fail (eg. insufficient buffer storage)

9.19.3.9 `gslc_ElemXListboxItemsSetGap()`

```
void gslc_ElemXListboxItemsSetGap (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nGap,
    gslc_tsColor colGap )
```

Configure the gap between listbox items.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nGap</i>	Set the gap between listbox items (0 for none)
in	<i>colGap</i>	Set the color of the gap between listbox items

## Returns

none

9.19.3.10 `gslc_ElemXListboxItemsSetSize()`

```
void gslc_ElemXListboxItemsSetSize (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemW,
    int16_t nItemH )
```

Configure the size of the listbox items.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nItemW</i>	Set the width of a listbox item (or -1 to auto-size)
in	<i>nItemH</i>	Set the height of a listbox item



**Returns**

none

**9.19.3.11 gslc\_ElemXListboxReset()**

```
void gslc_ElemXListboxReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Empty the listbox of all items.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update

**Returns**

none

**9.19.3.12 gslc\_ElemXListboxSetMargin()**

```
void gslc_ElemXListboxSetMargin (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nMarginW,
    int8_t nMarginH )
```

Configure the margin inside the listbox.

- Defines the region bewteen the element rect and the inner listbox items

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nMarginW</i>	Set the margin (horizontal) inside the listbox (0 for none)
in	<i>nMarginH</i>	Set the margin (horizontal) inside the listbox (0 for none)

**Returns**

none

### 9.19.3.13 gslc\_ElemXListboxSetScrollPos()

```
bool gslc_ElemXListboxSetScrollPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nScrollPos )
```

Set the Listbox scroll position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	Scroll the listbox so that the nScrollPos item is at the top (0 default)

#### Returns

true if success, false if fail

### 9.19.3.14 gslc\_ElemXListboxSetSel()

```
bool gslc_ElemXListboxSetSel (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nItemCurSel )
```

Set a Listbox element's current selection.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nItemCurSel</i>	Listbox item to select (or -1 for none)

#### Returns

true if success, false if fail

### 9.19.3.15 gslc\_ElemXListboxSetSelFunc()

```
void gslc_ElemXListboxSetSelFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XLISTBOX_SEL funcCb )
```

Assign the selection callback function for a Listbox.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to selection routine (or NULL for none)

## Returns

none

## 9.19.3.16 gslc\_ElemXListboxSetSize()

```
void gslc_ElemXListboxSetSize (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nRows,
    int8_t nCols )
```

Configure the number of rows & columns to display in the listbox.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Ptr to Element Reference to update
in	<i>nRows</i>	Number of rows (>= 1, or XLISTBOX_SIZE_AUTO to base on content)
in	<i>nCols</i>	Number of columns (>= 1)

## Returns

none

## 9.19.3.17 gslc\_ElemXListboxTouch()

```
bool gslc_ElemXListboxTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Listbox element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

## Parameters

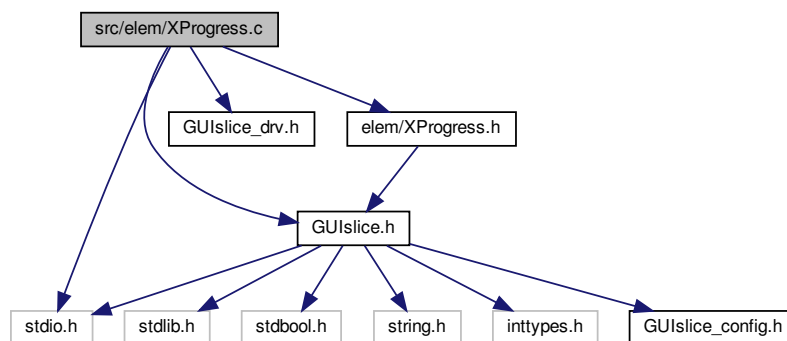
in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

## Returns

true if success, false otherwise

## 9.20 src/elem/XProgress.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XProgress.h"
#include <stdio.h>
Include dependency graph for XProgress.c:
```



## Functions

- `gslc_tsElemRef * gslc_ElemXProgressCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXProgress *pXData, gslc\_tsRect rElem, int16_t nMin, int16_t nMax, int16_t nVal, gslc\_tsColor colGauge, bool bVert)`  
*Create a Progress Bar Element.*
- `void gslc\_ElemXProgressSetVal (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nVal)`  
*Update a Gauge element's current value.*
- `void gslc\_ElemXProgressSetFlip (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bFlip)`  
*Set a Gauge element's fill direction.*
- `void gslc\_ElemXProgressSetGaugeCol (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colGauge)`  
*Set the gauge color.*
- `bool gslc\_ElemXProgressDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
*Draw a gauge element on the screen.*
- `bool gslc\_ElemXProgressDrawHelp (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_teRedrawType eRedraw)`  
*Helper function to draw a gauge with style: progress bar.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.20.1 Function Documentation

### 9.20.1.1 gslc\_ElemXProgressCreate()

```
gslc_tsElemRef* gslc_ElemXProgressCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXProgress * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )
```

Create a Progress Bar Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

#### Returns

Pointer to Element reference or NULL if failure

### 9.20.1.2 `gslc_ElemXProgressDraw()`

```
bool gslc_ElemXProgressDraw (
    void * pVGui,
    void * pVElemRef,
    gslc\_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

### 9.20.1.3 `gslc_ElemXProgressDrawHelp()`

```
bool gslc_ElemXProgressDrawHelp (
    gslc\_tsGui * pGui,
    gslc\_tsElemRef * pElemRef,
    gslc\_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: progress bar.

- Called from [gslc\\_ElemXProgressDraw\(\)](#)

#### Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

#### Returns

true if success, false otherwise

## 9.20.1.4 gslc\_ElemXProgressSetFlip()

```
void gslc_ElemXProgressSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )
```

Set a Gauge element's fill direction.

- Setting bFlip reverses the default fill direction
- Default fill direction for horizontal gauges: left-to-right
- Default fill direction for vertical gauges: bottom-to-top

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of fill from default

## Returns

none

## 9.20.1.5 gslc\_ElemXProgressSetGaugeCol()

```
void gslc_ElemXProgressSetGaugeCol (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge )
```

Set the gauge color.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colGauge</i>	Color for the gauge's fill

## Returns

none

### 9.20.1.6 gslc\_ElemXProgressSetVal()

```
void gslc_ElemXProgressSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

#### Returns

none

## 9.20.2 Variable Documentation

### 9.20.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

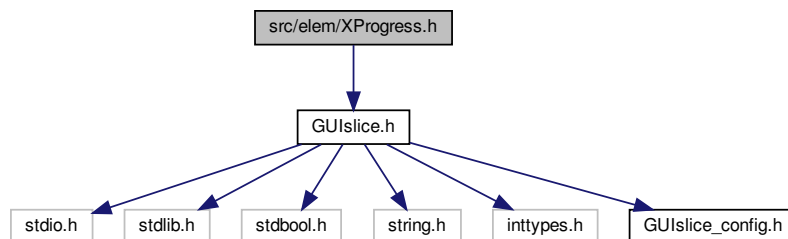
### 9.20.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

## 9.21 src/elem/XProgress.h File Reference

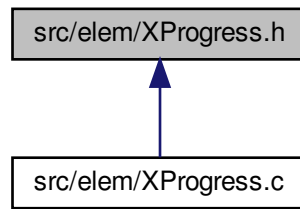
```
#include "GUIslice.h"
```

Include dependency graph for XProgress.h:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXProgress](#)  
*Extended data for Gauge element.*

## Macros

- #define [GSLC\\_TYPEX\\_PROGRESS](#)
- #define [gslc\\_ElemXProgressCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, nMin\_, nMax\_, nVal\_, col↵  
Frame\_, colFill\_, colGauge\_, bVert\_)  
*Create a Gauge Element in Flash.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXProgressCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsX↵  
Progress](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nMin, int16\_t nMax, int16\_t nVal, [gslc\\_tsColor](#) colGauge, bool  
bVert)  
*Create a Progress Bar Element.*
- void [gslc\\_ElemXProgressSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Update a Gauge element's current value.*
- void [gslc\\_ElemXProgressSetFlip](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFlip)  
*Set a Gauge element's fill direction.*
- void [gslc\\_ElemXProgressSetGaugeCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) col↵  
Gauge)  
*Set the gauge color.*
- bool [gslc\\_ElemXProgressDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a gauge element on the screen.*
- bool [gslc\\_ElemXProgressDrawHelp](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) e↵  
Redraw)  
*Helper function to draw a gauge with style: progress bar.*

### 9.21.1 Macro Definition Documentation

### 9.21.1.1 gslc\_ElemXProgressCreate\_P

```
#define gslc_ElemXProgressCreate_P(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    nMin_,
    nMax_,
    nVal_,
    colFrame_,
    colFill_,
    colGauge_,
    bVert_ )
```

Create a Gauge Element in Flash.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nMin_</i>	Minimum value of gauge for nVal comparison
in	<i>nMax_</i>	Maximum value of gauge for nVal comparison
in	<i>nVal_</i>	Starting value of gauge
in	<i>col↵ Frame_</i>	Color for the gauge frame
in	<i>colFill_</i>	Color for the gauge background fill
in	<i>col↵ Gauge_</i>	Color for the gauge indicator
in	<i>bVert_</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

#### Returns

none

### 9.21.1.2 GSLC\_TYPEX\_PROGRESS

```
#define GSLC_TYPEX_PROGRESS
```

## 9.21.2 Function Documentation

9.21.2.1 `gslc_ElemXProgressCreate()`

```

gslc_tsElemRef* gslc_ElemXProgressCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXProgress * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )

```

Create a Progress Bar Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

## Returns

Pointer to Element reference or NULL if failure

9.21.2.2 `gslc_ElemXProgressDraw()`

```

bool gslc_ElemXProgressDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.21.2.3 `gslc_ElemXProgressDrawHelp()`**

```
bool gslc_ElemXProgressDrawHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: progress bar.

- Called from [gslc\\_ElemXProgressDraw\(\)](#)

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

**Returns**

true if success, false otherwise

**9.21.2.4 `gslc_ElemXProgressSetFlip()`**

```
void gslc_ElemXProgressSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )
```

Set a Gauge element's fill direction.

- Setting `bFlip` reverses the default fill direction
- Default fill direction for horizontal gauges: left-to-right
- Default fill direction for vertical gauges: bottom-to-top

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of fill from default

## Returns

none

## 9.21.2.5 gslc\_ElemXProgressSetGaugeCol()

```
void gslc_ElemXProgressSetGaugeCol (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge )
```

Set the gauge color.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colGauge</i>	Color for the gauge's fill

## Returns

none

## 9.21.2.6 gslc\_ElemXProgressSetVal()

```
void gslc_ElemXProgressSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

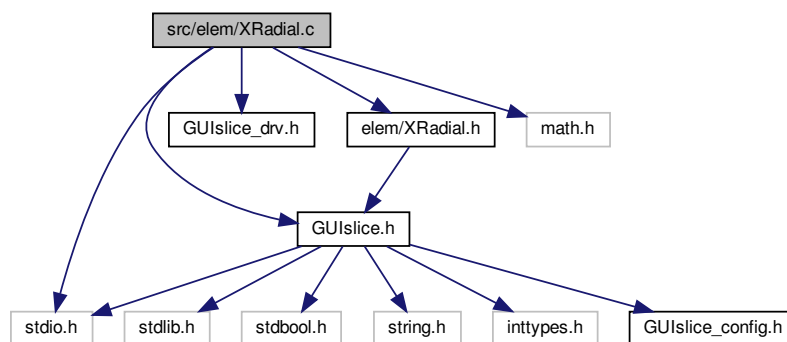
## Returns

none

## 9.22 src/elem/XRadial.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XRadial.h"
#include <stdio.h>
#include <math.h>
```

Include dependency graph for XRadial.c:



## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXRadialCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXRadial](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nMin, int16\_t nMax, int16\_t nVal, [gslc\\_tsColor](#) colGauge)  
*Create a Radial Gauge Element.*
- void [gslc\\_ElemXRadialSetIndicator](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colGauge, uint16\_t nIndicLen, uint16\_t nIndicTip, bool bIndicFill)  
*Configure the appearance of the Gauge indicator.*
- void [gslc\\_ElemXRadialSetTicks](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colTick, uint16\_t nTickCnt, uint16\_t nTickLen)  
*Configure the appearance of the Gauge ticks.*
- void [gslc\\_ElemXRadialSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Update a Gauge element's current value.*
- void [gslc\\_ElemXRadialSetFlip](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFlip)  
*Set a Gauge element's rotation direction.*
- bool [gslc\\_ElemXRadialDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a gauge element on the screen.*
- void [gslc\\_ElemXRadialDrawRadialHelp](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, uint16\_t nArrowLen, uint16\_t nArrowSz, int16\_t n64Ang, bool bFill, [gslc\\_tsColor](#) colFrame)
- bool [gslc\\_ElemXRadialDrawRadial](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Helper function to draw a gauge with style: radial.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.22.1 Function Documentation

### 9.22.1.1 gslc\_ElemXRadialCreate()

```
gslc_tsElemRef* gslc_ElemXRadialCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXRadial * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge )
```

Create a Radial Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator

#### Returns

Pointer to Element reference or NULL if failure

### 9.22.1.2 gslc\_ElemXRadialDraw()

```
bool gslc_ElemXRadialDraw (
    void * pvGui,
```

```
void * pvElemRef,
gslc_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

#### 9.22.1.3 [gslc\\_ElemXRadialDrawRadial\(\)](#)

```
bool gslc_ElemXRadialDrawRadial (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: radial.

- Called from [gslc\\_ElemXRadialDraw\(\)](#)

#### Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

#### Returns

true if success, false otherwise

#### 9.22.1.4 [gslc\\_ElemXRadialDrawRadialHelp\(\)](#)

```
void gslc_ElemXRadialDrawRadialHelp (
    gslc_tsGui * pGui,
```



```

    int16_t nX,
    int16_t nY,
    uint16_t nArrowLen,
    uint16_t nArrowSz,
    int16_t n64Ang,
    bool bFill,
    gslc_tsColor colFrame )

```

#### 9.22.1.5 gslc\_ElemXRadialSetFlip()

```

void gslc_ElemXRadialSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )

```

Set a Gauge element's rotation direction.

- Setting bFlip reverses the rotation direction
- Default rotation is clockwise. When bFlip is set, uses counter-clockwise

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of rotation from default

##### Returns

none

#### 9.22.1.6 gslc\_ElemXRadialSetIndicator()

```

void gslc_ElemXRadialSetIndicator (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge,
    uint16_t nIndicLen,
    uint16_t nIndicTip,
    bool bIndicFill )

```

Configure the appearance of the Gauge indicator.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Parameters**

in	<i>colGauge</i>	Color of the indicator
in	<i>nIndicLen</i>	Length of the indicator
in	<i>nIndicTip</i>	Size of the indicator tip
in	<i>bIndicFill</i>	Fill in the indicator if true

**Returns**

none

**9.22.1.7 gslc\_ElemXRadialSetTicks()**

```
void gslc_ElemXRadialSetTicks (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colTick,
    uint16_t nTickCnt,
    uint16_t nTickLen )
```

Configure the appearance of the Gauge ticks.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colTick</i>	Color of the gauge ticks
in	<i>nTickCnt</i>	Number of ticks to draw around / along gauge
in	<i>nTickLen</i>	Length of the tick marks to draw

**Returns**

none

**9.22.1.8 gslc\_ElemXRadialSetVal()**

```
void gslc_ElemXRadialSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

## Returns

none

## 9.22.2 Variable Documentation

## 9.22.2.1 ERRSTR\_NULL

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

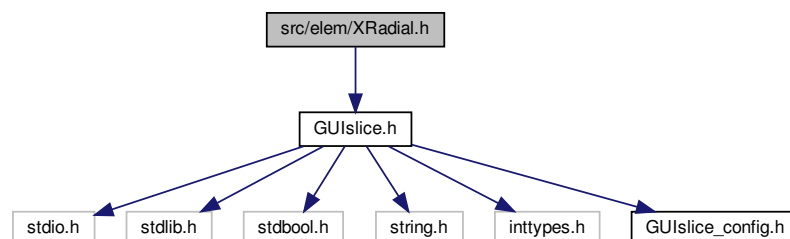
## 9.22.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```

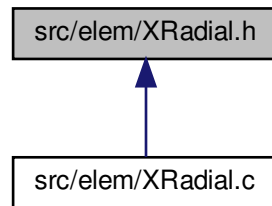
## 9.23 src/elem/XRadial.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XRadial.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXRadial](#)  
*Extended data for Gauge element.*

## Macros

- #define [GSLC\\_TYPEX\\_RADIAL](#)
- #define [gslc\\_ElemXRadialCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, nMin\_, nMax\_, nVal\_, col←  
Frame\_, colFill\_, colGauge\_)  
*Create a Gauge Element in Flash.*

## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXRadialCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsX←  
Radial](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nMin, int16\_t nMax, int16\_t nVal, [gslc\\_tsColor](#) colGauge)  
*Create a Radial Gauge Element.*
- void [gslc\\_ElemXRadialSetIndicator](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colGauge,  
uint16\_t nIndicLen, uint16\_t nIndicTip, bool bIndicFill)  
*Configure the appearance of the Gauge indicator.*
- void [gslc\\_ElemXRadialSetTicks](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colTick,  
uint16\_t nTickCnt, uint16\_t nTickLen)  
*Configure the appearance of the Gauge ticks.*
- void [gslc\\_ElemXRadialSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Update a Gauge element's current value.*
- void [gslc\\_ElemXRadialSetFlip](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFlip)  
*Set a Gauge element's rotation direction.*
- bool [gslc\\_ElemXRadialDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a gauge element on the screen.*
- bool [gslc\\_ElemXRadialDrawRadial](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) e←  
Redraw)  
*Helper function to draw a gauge with style: radial.*

## 9.23.1 Macro Definition Documentation

### 9.23.1.1 gslc\_ElemXRadialCreate\_P

```
#define gslc_ElemXRadialCreate_P(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    nMin_,
    nMax_,
    nVal_,
    colFrame_,
    colFill_,
    colGauge_ )
```

Create a Gauge Element in Flash.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nMin_</i>	Minimum value of gauge for nVal comparison
in	<i>nMax_</i>	Maximum value of gauge for nVal comparison
in	<i>nVal_</i>	Starting value of gauge
in	<i>col↵ Frame_</i>	Color for the gauge frame
in	<i>colFill_</i>	Color for the gauge background fill
in	<i>col↵ Gauge_</i>	Color for the gauge indicator

#### Returns

none

### 9.23.1.2 GSLC\_TYPEX\_RADIAL

```
#define GSLC_TYPEX_RADIAL
```

## 9.23.2 Function Documentation

### 9.23.2.1 `gslc_ElemXRadialCreate()`

```
gslc_tsElemRef* gslc_ElemXRadialCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXRadial * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge )
```

Create a Radial Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator

#### Returns

Pointer to Element reference or NULL if failure

### 9.23.2.2 `gslc_ElemXRadialDraw()`

```
bool gslc_ElemXRadialDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from `gslc_ElemDraw()`

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.23.2.3 `gslc_ElemXRadialDrawRadial()`

```
bool gslc_ElemXRadialDrawRadial (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: radial.

- Called from [gslc\\_ElemXRadialDraw\(\)](#)

## Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

## Returns

true if success, false otherwise

9.23.2.4 `gslc_ElemXRadialSetFlip()`

```
void gslc_ElemXRadialSetFlip (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bFlip )
```

Set a Gauge element's rotation direction.

- Setting `bFlip` reverses the rotation direction
- Default rotation is clockwise. When `bFlip` is set, uses counter-clockwise

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bFlip</i>	If set, reverse direction of rotation from default

**Returns**

none

**9.23.2.5 gslc\_ElemXRadialSetIndicator()**

```
void gslc_ElemXRadialSetIndicator (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colGauge,
    uint16_t nIndicLen,
    uint16_t nIndicTip,
    bool bIndicFill )
```

Configure the appearance of the Gauge indicator.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colGauge</i>	Color of the indicator
in	<i>nIndicLen</i>	Length of the indicator
in	<i>nIndicTip</i>	Size of the indicator tip
in	<i>bIndicFill</i>	Fill in the indicator if true

**Returns**

none

**9.23.2.6 gslc\_ElemXRadialSetTicks()**

```
void gslc_ElemXRadialSetTicks (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colTick,
    uint16_t nTickCnt,
    uint16_t nTickLen )
```

Configure the appearance of the Gauge ticks.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colTick</i>	Color of the gauge ticks
in	<i>nTickCnt</i>	Number of ticks to draw around / along gauge
in	<i>nTickLen</i>	Length of the tick marks to draw

## Returns

none

## 9.23.2.7 gslc\_ElemXRadialSetVal()

```
void gslc_ElemXRadialSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

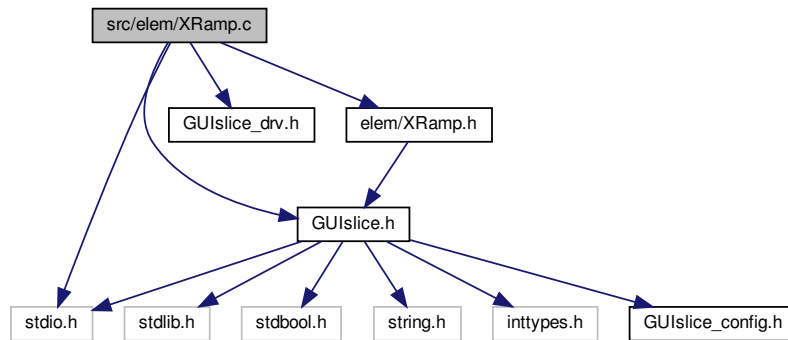
## Returns

none

## 9.24 src/elem/XRamp.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XRamp.h"
#include <stdio.h>
```

Include dependency graph for XRamp.c:



## Functions

- `gslc_tsElemRef * gslc_ElemXRampCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXRamp *pXData, gslc\_tsRect rElem, int16_t nMin, int16_t nMax, int16_t nVal, gslc\_tsColor colGauge, bool bVert)`  
*Create a Ramp Gauge Element.*
- `void gslc\_ElemXRampSetVal (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nVal)`  
*Update a Gauge element's current value.*
- `bool gslc\_ElemXRampDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
*Draw a gauge element on the screen.*
- `bool gslc\_ElemXRampDrawHelp (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_teRedrawType eRedraw)`  
*Helper function to draw a gauge with style: ramp.*

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

### 9.24.1 Function Documentation

#### 9.24.1.1 `gslc_ElemXRampCreate()`

```

gslc_tsElemRef* gslc_ElemXRampCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc\_tsXRamp * pXData,
    gslc\_tsRect rElem,
    int16_t nMin,

```

```

    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )

```

Create a Ramp Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

#### Returns

Pointer to Element reference or NULL if failure

#### 9.24.1.2 gslc\_ElemXRampDraw()

```

bool gslc_ElemXRampDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.24.1.3 gslc\_ElemXRampDrawHelp()**

```
bool gslc_ElemXRampDrawHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: ramp.

- Called from [gslc\\_ElemXRampDraw\(\)](#)

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

**Returns**

true if success, false otherwise

**9.24.1.4 gslc\_ElemXRampSetVal()**

```
void gslc_ElemXRampSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in create()

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

### Returns

none

## 9.24.2 Variable Documentation

### 9.24.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

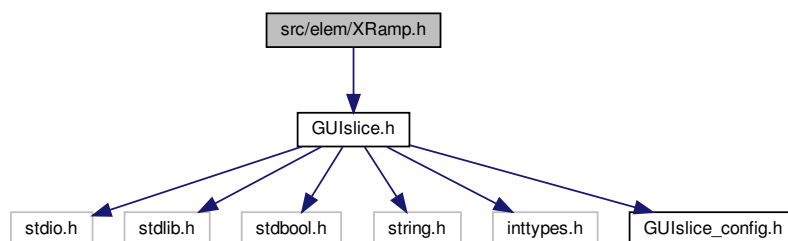
### 9.24.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

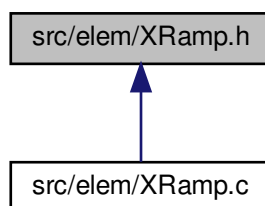
## 9.25 src/elem/XRamp.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XRamp.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXRamp](#)  
*Extended data for Gauge element.*

## Macros

- #define [GSLC\\_TYPEX\\_RAMP](#)
- #define [gslc\\_ElemXRampCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, nMin\_, nMax\_, nVal\_, colFrame\_, colFill\_)  
*Create a Gauge Element in Flash.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXRampCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXRamp](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nMin, int16\_t nMax, int16\_t nVal, [gslc\\_tsColor](#) colGauge, bool bVert)  
*Create a Ramp Gauge Element.*
- void [gslc\\_ElemXRampSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)  
*Update a Gauge element's current value.*
- bool [gslc\\_ElemXRampDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a gauge element on the screen.*
- bool [gslc\\_ElemXRampDrawHelp](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Helper function to draw a gauge with style: ramp.*

### 9.25.1 Macro Definition Documentation

#### 9.25.1.1 [gslc\\_ElemXRampCreate\\_P](#)

```
#define gslc_ElemXRampCreate_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    nMin_,  
    nMax_,  
    nVal_,  
    colFrame_,  
    colFill_ )
```

Create a Gauge Element in Flash.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nMin_</i>	Minimum value of gauge for nVal comparison
in	<i>nMax_</i>	Maximum value of gauge for nVal comparison
in	<i>nVal_</i>	Starting value of gauge
in	<i>colFrame_</i>	Color for the gauge frame
in	<i>colFill_</i>	Color for the gauge background fill

## Returns

none

## 9.25.1.2 GSLC\_TYPEX\_RAMP

```
#define GSLC_TYPEX_RAMP
```

## 9.25.2 Function Documentation

## 9.25.2.1 gslc\_ElemXRampCreate()

```
gslc_tsElemRef* gslc_ElemXRampCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXRamp * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    gslc_tsColor colGauge,
    bool bVert )
```

Create a Ramp Gauge Element.

- Draws a gauge element that represents a proportion (nVal) between nMin and nMax.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining gauge size
in	<i>nMin</i>	Minimum value of gauge for nVal comparison
in	<i>nMax</i>	Maximum value of gauge for nVal comparison
in	<i>nVal</i>	Starting value of gauge
in	<i>colGauge</i>	Color for the gauge indicator
in	<i>bVert</i>	Flag to indicate vertical vs horizontal action (true = vertical, false = horizontal)

## Returns

Pointer to Element reference or NULL if failure

9.25.2.2 `gslc_ElemXRampDraw()`

```
bool gslc_ElemXRampDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a gauge element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.25.2.3 `gslc_ElemXRampDrawHelp()`

```
bool gslc_ElemXRampDrawHelp (
    gslc_tsGui * pGui,
```



```
gslc_tsElemRef * pElemRef,
gslc_teRedrawType eRedraw )
```

Helper function to draw a gauge with style: ramp.

- Called from `gslc_ElemXRampDraw()`

#### Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pElemRef</i>	Ptr to Element reference
in	<i>eRedraw</i>	Redraw status

#### Returns

true if success, false otherwise

#### 9.25.2.4 gslc\_ElemXRampSetVal()

```
void gslc_ElemXRampSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Update a Gauge element's current value.

- Note that min & max values are assigned in `create()`

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

#### Returns

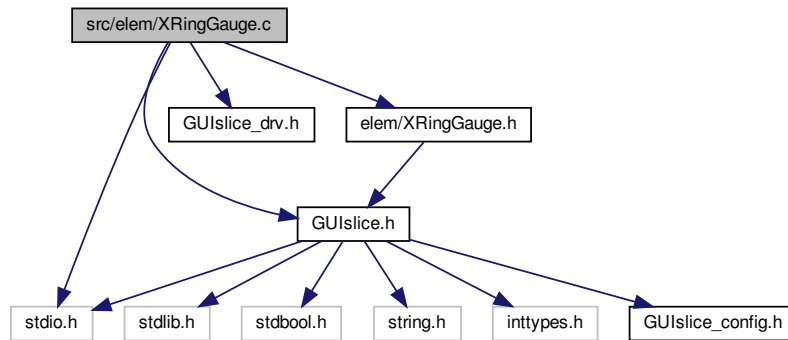
none

## 9.26 src/elem/XRingGauge.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XRingGauge.h"
```

```
#include <stdio.h>
```

Include dependency graph for XRingGauge.c:



## Functions

- `gslc_tsElemRef * gslc_ElemXRingGaugeCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXRingGauge *pXData, gslc\_tsRect rElem, char *pStrBuf, uint8_t nStrBufMax, int16_t nFontId)`  
*Create an XRingGauge element.*
- `bool gslc\_ElemXRingGaugeDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
*Draw the template element on the screen.*
- `void gslc\_ElemXRingGaugeSetVal (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nVal)`  
*Set an Ring Gauge current indicator value.*
- `void gslc\_ElemXRingGaugeSetValRange (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nValMin, int16_t nValMax)`  
*Defines the range of values that may be passed into SetVal(), used to scale the input to SetVal().*
- `void gslc\_ElemXRingGaugeSetAngleRange (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nStart, int16_t nRange, bool bClockwise)`  
*Defines the angular range of the gauge, including both the active and inactive regions.*
- `void gslc\_ElemXRingGaugeSetThickness (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int8_t nThickness)`  
*Defines the thickness of the ring arcs.*
- `void gslc\_ElemXRingGaugeSetColorActiveFlat (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colActive)`  
*Defines the color of the active region to be a flat (constant) color.*
- `void gslc\_ElemXRingGaugeSetColorActiveGradient (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colStart, gslc\_tsColor colEnd)`  
*Defines the color of the active region to be a gradient using two color stops.*
- `void gslc\_ElemXRingGaugeSetColorInactive (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor colInactive)`  
*Defines the color of the inactive region to be a flat (constant) color.*
- `void gslc\_ElemXRingGaugeSetQuality (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, uint16_t nSegments)`  
*Sets the quality of the ring drawing by defining the number of segments that are used when rendering a 360 degree gauge. The larger the number, the more segments are used and the smoother the curve.*

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

## 9.26.1 Function Documentation

### 9.26.1.1 gslc\_ElemXRingGaugeCreate()

```
gslc_tsElemRef* gslc_ElemXRingGaugeCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXRingGauge * pXData,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )
```

Create an XRingGauge element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	The square box that bounds the ring element. If a rectangular region is provided, then the ring control will be centered in the long axis.
in	<i>pStrBuf</i>	String buffer to use for gauge inner text
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf)
in	<i>nFontId</i>	Font ID to use for text display

#### Returns

Pointer to Element reference or NULL if failure

### 9.26.1.2 gslc\_ElemXRingGaugeDraw()

```
bool gslc_ElemXRingGaugeDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw the template element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.26.1.3 `gslc_ElemXRingGaugeSetAngleRange()`

```
void gslc_ElemXRingGaugeSetAngleRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nStart,
    int16_t nRange,
    bool bClockwise )
```

Defines the angular range of the gauge, including both the active and inactive regions.

- `nStart` defines the angle at the beginning of the active region.
- The current position marks the end of the active region and the beginning of the inactive region.
- `nRange` defines the angular range from the start of the active region to the end of the inactive region. In most cases, a range of 360 degrees is used.
- All angles are measured in units of degrees.
- Angles are measured with 0 at the top, 90 towards the right, 180 towards the bottom, 270 towards the left, etc.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nStart</i>	Define angle of start of active region (measured in degrees)
in	<i>nRange</i>	Define angular range from strt of active region to end of the inactive region (measured in degrees)
in	<i>bClockwise</i>	Defines the direction in which the active region grows (true for clockwise) [FORCED TRUE, FOR FUTURE IMPLEMENTATION]

## Returns

none

## 9.26.1.4 gslc\_ElemXRingGaugeSetColorActiveFlat()

```
void gslc_ElemXRingGaugeSetColorActiveFlat (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colActive )
```

Defines the color of the active region to be a flat (constant) color.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colActive</i>	Color of active region

## Returns

none

## 9.26.1.5 gslc\_ElemXRingGaugeSetColorActiveGradient()

```
void gslc_ElemXRingGaugeSetColorActiveGradient (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colStart,
    gslc_tsColor colEnd )
```

Defines the color of the active region to be a gradient using two color stops.

The active region will be filled according to the proportion between nMin and nMax. The gradient is defined by a linear RGB blend between the two color stops(colStart and colEnd)

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colStart</i>	Starting color of gradient fill
in	<i>colEnd</i>	Ending color of gradient fill

## Returns

none

## 9.26.1.6 gslc\_ElemXRingGaugeSetColorInactive()

```
void gslc_ElemXRingGaugeSetColorInactive (
    gslc_tsGui * pGui,
```

```
gslc_tsElemRef * pElemRef,
gslc_tsColor colInactive )
```

Defines the color of the inactive region to be a flat (constant) color.

The inactive color is often set to be the same as the background but it can be set to a different color to indicate the remainder of the value range that is yet to be filled.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colInactive</i>	Color of inactive region

#### Returns

none

#### 9.26.1.7 gslc\_ElemXRingGaugeSetQuality()

```
void gslc_ElemXRingGaugeSetQuality (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nSegments )
```

Sets the quality of the ring drawing by defining the number of segments that are used when rendering a 360 degree gauge. The larger the number, the more segments are used and the smoother the curve.

A larger ring gauge may need a higher quality number to maintain a smoothed curve appearance.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nSegments</i>	Number of arc segments to render a complete circle. The higher the value, the smoother the ring. Note that 360/nSegments should be an integer result, thus the allowable quality settings are: 360 (max quality), 180, 120, 90, 72, 60, 45, 40, 36 (low quality), etc.

#### Returns

none

#### 9.26.1.8 gslc\_ElemXRingGaugeSetThickness()

```
void gslc_ElemXRingGaugeSetThickness (
    gslc_tsGui * pGui,
```

```
gslc_tsElemRef * pElemRef,
int8_t nThickness )
```

Defines the thickness of the ring arcs.

More specifically, it defines the reduction in radius from the outer radius to the inner radius in pixels.

- Default thickness is 10 pixels

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nThickness</i>	Thickness of ring

#### Returns

none

#### 9.26.1.9 gslc\_ElemXRingGaugeSetVal()

```
void gslc_ElemXRingGaugeSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Set an Ring Gauge current indicator value.

Updates the current value of the ring gauge. The active region will be drawn up to the position defined by nVal within the value range defined by SetValRange(nMin,nMax). A SetVal() close to nMin will cause a very small active region to be drawn and a large remainder drawn in the inactive color, whereas a SetVal() close to nMax will cause a more complete active region to be drawn. When SetVal() equals nMax, the entire angular range will be drawn in the active color (and no inactive region).

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New position value

#### Returns

none

#### 9.26.1.10 gslc\_ElemXRingGaugeSetValRange()

```
void gslc_ElemXRingGaugeSetValRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nValMin,
    int16_t nValMax )
```

Defines the range of values that may be passed into SetVal(), used to scale the input to SetVal().

- Default is 0..100.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nValMin</i>	Minimum value
in	<i>nValMax</i>	Maximum value

##### Returns

none

### 9.26.2 Variable Documentation

#### 9.26.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

#### 9.26.2.2 ERRSTR\_PXD\_NULL

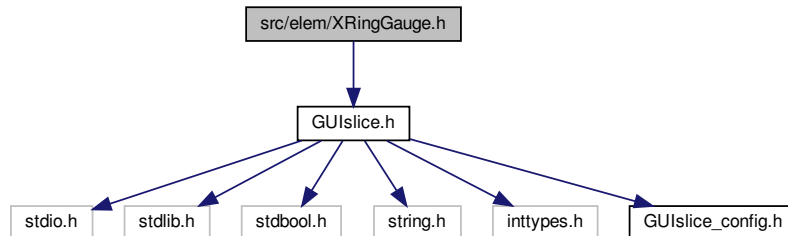
```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```



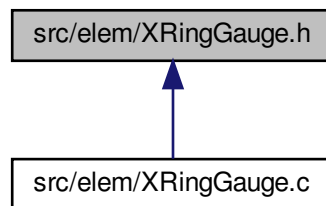
## 9.27 src/elem/XRingGauge.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XRingGauge.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gslc\\_tsXRingGauge](#)  
*Extended data for XRingGauge element.*

### Macros

- #define [GSLC\\_TYPEX\\_RING](#)
- #define [XRING\\_STR\\_MAX](#)

### Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXRingGaugeCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXRingGauge](#) \*pXData, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create an XRingGauge element.*
- bool [gslc\\_ElemXRingGaugeDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw the template element on the screen.*

- void [gslc\\_ElemXRingGaugeSetVal](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nVal)

*Set an Ring Gauge current indicator value.*

- void [gslc\\_ElemXRingGaugeSetAngleRange](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nStart, int16\_t nRange, bool bClockwise)

*Defines the angular range of the gauge, including both the active and inactive regions.*

- void [gslc\\_ElemXRingGaugeSetValRange](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nValMin, int16\_t nValMax)

*Defines the range of values that may be passed into SetVal(), used to scale the input to SetVal().*

- void [gslc\\_ElemXRingGaugeSetThickness](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nThickness)

*Defines the thickness of the ring arcs.*

- void [gslc\\_ElemXRingGaugeSetQuality](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint16\_t nSegments)

*Sets the quality of the ring drawing by defining the number of segments that are used when rendering a 360 degree gauge. The larger the number, the more segments are used and the smoother the curve.*

- void [gslc\\_ElemXRingGaugeSetColorInactive](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colInactive)

*Defines the color of the inactive region to be a flat (constant) color.*

- void [gslc\\_ElemXRingGaugeSetColorActiveFlat](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colActive)

*Defines the color of the active region to be a flat (constant) color.*

- void [gslc\\_ElemXRingGaugeSetColorActiveGradient](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colStart, [gslc\\_tsColor](#) colEnd)

*Defines the color of the active region to be a gradient using two color stops.*

## 9.27.1 Macro Definition Documentation

### 9.27.1.1 GSLC\_TYPEX\_RING

```
#define GSLC_TYPEX_RING
```

### 9.27.1.2 XRING\_STR\_MAX

```
#define XRING_STR_MAX
```

## 9.27.2 Function Documentation

### 9.27.2.1 gslc\_ElemXRingGaugeCreate()

```
gslc\_tsElemRef* gslc\_ElemXRingGaugeCreate (
    gslc\_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc\_tsXRingGauge * pXData,
    gslc\_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )
```

Create an XRingGauge element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	The square box that bounds the ring element. If a rectangular region is provided, then the ring control will be centered in the long axis.
in	<i>pStrBuf</i>	String buffer to use for gauge inner text
in	<i>nStrBufMax</i>	Maximum length of string buffer (pStrBuf)
in	<i>nFontId</i>	Font ID to use for text display

## Returns

Pointer to Element reference or NULL if failure

## 9.27.2.2 gslc\_ElemXRingGaugeDraw()

```
bool gslc_ElemXRingGaugeDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw the template element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

## 9.27.2.3 gslc\_ElemXRingGaugeSetAngleRange()

```
void gslc_ElemXRingGaugeSetAngleRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nStart,
```

```

    int16_t nRange,
    bool bClockwise )

```

Defines the angular range of the gauge, including both the active and inactive regions.

- nStart defines the angle at the beginning of the active region.
- The current position marks the end of the active region and the beginning of the inactive region.
- nRange defines the angular range from the start of the active region to the end of the inactive region. In most cases, a range of 360 degrees is used.
- All angles are measured in units of degrees.
- Angles are measured with 0 at the top, 90 towards the right, 180 towards the bottom, 270 towards the left, etc.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nStart</i>	Define angle of start of active region (measured in degrees)
in	<i>nRange</i>	Define angular range from strt of active region to end of the inactive region (measured in degrees)
in	<i>bClockwise</i>	Defines the direction in which the active region grows (true for clockwise) [FORCED TRUE, FOR FUTURE IMPLEMENTATION]

#### Returns

none

#### 9.27.2.4 gslc\_ElemXRingGaugeSetColorActiveFlat()

```

void gslc_ElemXRingGaugeSetColorActiveFlat (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colActive )

```

Defines the color of the active region to be a flat (constant) color.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colActive</i>	Color of active region

#### Returns

none

## 9.27.2.5 gslc\_ElemXRingGaugeSetColorActiveGradient()

```
void gslc_ElemXRingGaugeSetColorActiveGradient (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colStart,
    gslc_tsColor colEnd )
```

Defines the color of the active region to be a gradient using two color stops.

The active region will be filled according to the proportion between nMin and nMax. The gradient is defined by a linear RGB blend between the two color stops(colStart and colEnd)

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colStart</i>	Starting color of gradient fill
in	<i>colEnd</i>	Ending color of gradient fill

## Returns

none

## 9.27.2.6 gslc\_ElemXRingGaugeSetColorInactive()

```
void gslc_ElemXRingGaugeSetColorInactive (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor colInactive )
```

Defines the color of the inactive region to be a flat (constant) color.

The inactive color is often set to be the same as the background but it can be set to a different color to indicate the remainder of the value range that is yet to be filled.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>colInactive</i>	Color of inactive region

## Returns

none

### 9.27.2.7 gslc\_ElemXRingGaugeSetQuality()

```
void gslc_ElemXRingGaugeSetQuality (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint16_t nSegments )
```

Sets the quality of the ring drawing by defining the number of segments that are used when rendering a 360 degree gauge. The larger the number, the more segments are used and the smoother the curve.

A larger ring gauge may need a higher quality number to maintain a smoothed curve appearance.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nSegments</i>	Number of arc segments to render a complete circle. The higher the value, the smoother the ring. Note that 360/nSegments should be an integer result, thus the allowable quality settings are: 360 (max quality), 180, 120, 90, 72, 60, 45, 40, 36 (low quality), etc.

#### Returns

none

### 9.27.2.8 gslc\_ElemXRingGaugeSetThickness()

```
void gslc_ElemXRingGaugeSetThickness (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int8_t nThickness )
```

Defines the thickness of the ring arcs.

More specifically, it defines the reduction in radius from the outer radius to the inner radius in pixels.

- Default thickness is 10 pixels

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nThickness</i>	Thickness of ring

#### Returns

none

## 9.27.2.9 gslc\_ElemXRingGaugeSetVal()

```
void gslc_ElemXRingGaugeSetVal (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nVal )
```

Set an Ring Gauge current indicator value.

Updates the current value of the ring gauge. The active region will be drawn up to the position defined by nVal within the value range defined by SetValRange(nMin,nMax). A SetVal() close to nMin will cause a very small active region to be drawn and a large remainder drawn in the inactive color, whereas a SetVal() close to nMax will cause a more complete active region to be drawn. When SetVal() equals nMax, the entire angular range will be drawn in the active color (and no inactive region).

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New position value

## Returns

none

## 9.27.2.10 gslc\_ElemXRingGaugeSetValRange()

```
void gslc_ElemXRingGaugeSetValRange (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nValMin,
    int16_t nValMax )
```

Defines the range of values that may be passed into SetVal(), used to scale the input to SetVal().

- Default is 0..100.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nValMin</i>	Minimum value
in	<i>nValMax</i>	Maximum value

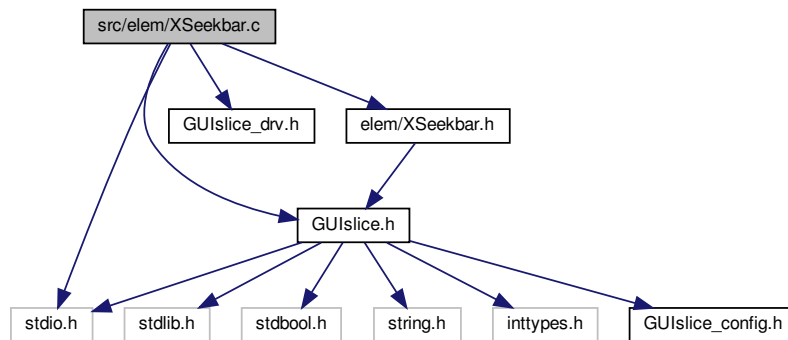
## Returns

none

## 9.28 src/elem/XSeekBar.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XSeekBar.h"
#include <stdio.h>
```

Include dependency graph for XSeekBar.c:



### Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXSeekBarCreate](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nElemId, [int16\\_t](#) nPage, [gslc\\_tsXSeekBar](#) \*pXData, [gslc\\_tsRect](#) rElem, [int16\\_t](#) nPosMin, [int16\\_t](#) nPosMax, [int16\\_t](#) nPos, [uint8\\_t](#) nProgressW, [uint8\\_t](#) nRemainW, [uint8\\_t](#) nThumbSz, [gslc\\_tsColor](#) colProgress, [gslc\\_tsColor](#) colRemain, [gslc\\_tsColor](#) colThumb, [bool](#) bVert)  
*Create a Seekbar Element.*
- void [gslc\\_ElemXSeekBarSetStyle](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [bool](#) bTrimThumb, [gslc\\_tsColor](#) colTrim, [bool](#) bFrameThumb, [gslc\\_tsColor](#) colFrame, [uint16\\_t](#) nTickDiv, [int16\\_t](#) nTickLen, [gslc\\_tsColor](#) colTick)  
*Set a Seekbar element's style, this includes thumb customizations and tick marks.*
- [int](#) [gslc\\_ElemXSeekBarGetPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a Seekbar element's current position.*
- void [gslc\\_ElemXSeekBarSetPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [int16\\_t](#) nPos)  
*Set a Seekbar element's current position.*
- void [gslc\\_ElemXSeekBarSetPosFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_XSEEKBAR\\_POS](#) funcCb)  
*Assign the position callback function for a slider.*
- [bool](#) [gslc\\_ElemXSeekBarDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Seekbar element on the screen.*
- [bool](#) [gslc\\_ElemXSeekBarTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, [int16\\_t](#) nRelX, [int16\\_t](#) nRelY)  
*Handle touch events to Seekbar element.*

### Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []



## 9.28.1 Function Documentation

### 9.28.1.1 gslc\_ElemXSeekBarCreate()

```
gslc_tsElemRef* gslc_ElemXSeekBarCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSeekBar * pXData,
    gslc_tsRect rElem,
    int16_t nPosMin,
    int16_t nPosMax,
    int16_t nPos,
    uint8_t nProgressW,
    uint8_t nRemainW,
    uint8_t nThumbSz,
    gslc_tsColor colProgress,
    gslc_tsColor colRemain,
    gslc_tsColor colThumb,
    bool bVert )
```

Create a Seekbar Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nPosMin</i>	Minimum position value
in	<i>nPosMax</i>	Maximum position value
in	<i>nPos</i>	Starting position value
in	<i>nProgressW</i>	Width of progress track
in	<i>nRemainW</i>	Width of remaining track
in	<i>nThumbSz</i>	Size of the thumb control
in	<i>colProgress</i>	Color of progress fill bar
in	<i>colRemain</i>	Color remaining fill bar
in	<i>colThumb</i>	Color for the thumb indicator
in	<i>bVert</i>	Orientation (true for vertical)

#### Returns

Pointer to Element reference or NULL if failure

### 9.28.1.2 `gslc_ElemXSeekBarDraw()`

```
bool gslc_ElemXSeekBarDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Seekbar element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

### 9.28.1.3 `gslc_ElemXSeekBarGetPos()`

```
int gslc_ElemXSeekBarGetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Seekbar element's current position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

Current slider position

### 9.28.1.4 `gslc_ElemXSeekBarSetPos()`

```
void gslc_ElemXSeekBarSetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nPos )
```

Set a Seekbar element's current position.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nPos</i>	New position value

## Returns

none

## 9.28.1.5 gslc\_ElemXSeekBarSetPosFunc()

```
void gslc_ElemXSeekBarSetPosFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XSEEKBAR_POS funcCb )
```

Assign the position callback function for a slider.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to position routine (or NULL for none)

## Returns

none

## 9.28.1.6 gslc\_ElemXSeekBarSetStyle()

```
void gslc_ElemXSeekBarSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bTrimThumb,
    gslc_tsColor colTrim,
    bool bFrameThumb,
    gslc_tsColor colFrame,
    uint16_t nTickDiv,
    int16_t nTickLen,
    gslc_tsColor colTick )
```

Set a Seekbar element's style, this includes thumb customizations and tick marks.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bTrimThumb</i>	Show a colored trim for thumb?
in	<i>colTrim</i>	Color of thumb trim
in	<i>bFrameThumb</i>	Show a frame around thumb?
in	<i>colFrame</i>	Color of thumb frame
in	<i>nTickDiv</i>	Number of tick divisions to show (0 for none)
in	<i>nTickLen</i>	Length of tick marks
in	<i>colTick</i>	Color of ticks

**Returns**

none

**9.28.1.7 gslc\_ElemXSeekBarTouch()**

```
bool gslc_ElemXSeekBarTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Seekbar element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

**Returns**

true if success, false otherwise

**9.28.2 Variable Documentation**

## 9.28.2.1 ERRSTR\_NULL

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

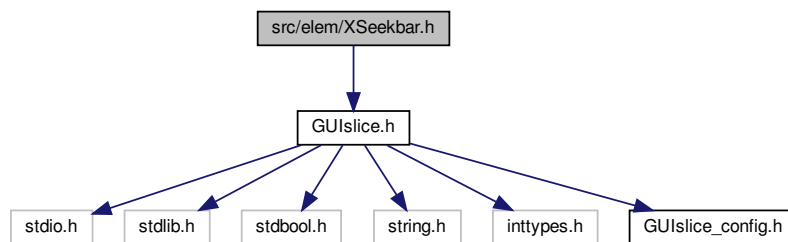
## 9.28.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```

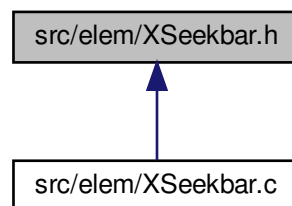
## 9.29 src/elem/XSeekBar.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XSeekBar.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXSeekBar](#)

*Extended data for Seekbar element.*

## Macros

- `#define GSLC_TYPEX_SEEKBAR`
- `#define gslc_ElemXSeekBarCreate_P(pGui, nElemId, nPage, nX, nY, nW, nH, nPosMin_, nPosMax_, nPos↵  
_, nProgressW_, nRemainW_, nThumbSz_, colProgress_, colRemain_, colThumb_, bVert_, colFrame_↵  
, colFill_)`

*Create a Seekbar Element in Flash.*

## Typedefs

- `typedef bool(* GSLC_CB_XSEEKBAR_POS) (void *pvGui, void *pvElem, int16_t nPos)`

*Callback function for slider feedback.*

## Functions

- `gslc_tsElemRef * gslc_ElemXSeekBarCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsX↵  
SeekBar *pXData, gslc_tsRect rElem, int16_t nPosMin, int16_t nPosMax, int16_t nPos, uint8_t nProgressW,  
uint8_t nRemainW, uint8_t nThumbSz, gslc_tsColor colProgress, gslc_tsColor colRemain, gslc_tsColor col↵  
Thumb, bool bVert)`

*Create a Seekbar Element.*

- `void gslc_ElemXSeekBarSetStyle (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bTrimThumb, gslc_↵  
tsColor colTrim, bool bFrameThumb, gslc_tsColor colFrame, uint16_t nTickDiv, int16_t nTickLen, gslc_tsColor  
colTick)`

*Set a Seekbar element's style, this includes thumb customizations and tick marks.*

- `int gslc_ElemXSeekBarGetPos (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`

*Get a Seekbar element's current position.*

- `void gslc_ElemXSeekBarSetPos (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nPos)`

*Set a Seekbar element's current position.*

- `void gslc_ElemXSeekBarSetPosFunc (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XSEEK↵  
BAR_POS funcCb)`

*Assign the position callback function for a slider.*

- `bool gslc_ElemXSeekBarDraw (void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`

*Draw a Seekbar element on the screen.*

- `bool gslc_ElemXSeekBarTouch (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_↵  
_t nRelY)`

*Handle touch events to Seekbar element.*

### 9.29.1 Macro Definition Documentation

## 9.29.1.1 gslc\_ElemXSeekBarCreate\_P

```

#define gslc_ElemXSeekBarCreate_P(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    nPosMin_,
    nPosMax_,
    nPos_,
    nProgressW_,
    nRemainW_,
    nThumbSz_,
    colProgress_,
    colRemain_,
    colThumb_,
    bVert_,
    colFrame_,
    colFill_ )

```

Create a Seekbar Element in Flash.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nPosMin_</i>	Minimum position value
in	<i>nPosMax_</i>	Maximum position value
in	<i>nPos_</i>	Starting position value
in	<i>nProgress↔ W_</i>	Width of progress track
in	<i>nRemainW↔ _</i>	Width of remaining track
in	<i>nThumbSz↔ _</i>	Size of the thumb control
in	<i>col↔ Progress_</i>	Color of progress fill bar
in	<i>colRemain↔ _</i>	Color remaining fill bar
in	<i>colThumb_</i>	Color for the thumb indicator
in	<i>bVert_</i>	Orientation (true for vertical)
in	<i>colFrame_</i>	Color of the element frame
in	<i>colFill_</i>	Color of the element fill

**Returns**

none

**9.29.1.2 GSLC\_TYPEX\_SEEKBAR**

```
#define GSLC_TYPEX_SEEKBAR
```

**9.29.2 Typedef Documentation****9.29.2.1 GSLC\_CB\_XSEEKBAR\_POS**

```
typedef bool(* GSLC_CB_XSEEKBAR_POS) (void *pvGui, void *pvElem, int16_t nPos)
```

Callback function for slider feedback.

**9.29.3 Function Documentation****9.29.3.1 gslc\_ElemXSeekbarCreate()**

```
gslc_tsElemRef* gslc_ElemXSeekbarCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSeekbar * pXData,
    gslc_tsRect rElem,
    int16_t nPosMin,
    int16_t nPosMax,
    int16_t nPos,
    uint8_t nProgressW,
    uint8_t nRemainW,
    uint8_t nThumbSz,
    gslc_tsColor colProgress,
    gslc_tsColor colRemain,
    gslc_tsColor colThumb,
    bool bVert )
```

Create a Seekbar Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)



## Parameters

in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nPosMin</i>	Minimum position value
in	<i>nPosMax</i>	Maximum position value
in	<i>nPos</i>	Starting position value
in	<i>nProgressW</i>	Width of progress track
in	<i>nRemainW</i>	Width of remaining track
in	<i>nThumbSz</i>	Size of the thumb control
in	<i>colProgress</i>	Color of progress fill bar
in	<i>colRemain</i>	Color remaining fill bar
in	<i>colThumb</i>	Color for the thumb indicator
in	<i>bVert</i>	Orientation (true for vertical)

## Returns

Pointer to Element reference or NULL if failure

9.29.3.2 `gslc_ElemXSeekBarDraw()`

```
bool gslc_ElemXSeekBarDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Seekbar element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

### 9.29.3.3 gslc\_ElemXSeekBarGetPos()

```
int gslc_ElemXSeekBarGetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Seekbar element's current position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

Current slider position

### 9.29.3.4 gslc\_ElemXSeekBarSetPos()

```
void gslc_ElemXSeekBarSetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nPos )
```

Set a Seekbar element's current position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nPos</i>	New position value

#### Returns

none

### 9.29.3.5 gslc\_ElemXSeekBarSetPosFunc()

```
void gslc_ElemXSeekBarSetPosFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XSEEKBAR_POS funcCb )
```

Assign the position callback function for a slider.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to position routine (or NULL for none)

## Returns

none

## 9.29.3.6 gslc\_ElemXSeekBarSetStyle()

```
void gslc_ElemXSeekBarSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bTrimThumb,
    gslc_tsColor colTrim,
    bool bFrameThumb,
    gslc_tsColor colFrame,
    uint16_t nTickDiv,
    int16_t nTickLen,
    gslc_tsColor colTick )
```

Set a Seekbar element's style, this includes thumb customizations and tick marks.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bTrimThumb</i>	Show a colored trim for thumb?
in	<i>colTrim</i>	Color of thumb trim
in	<i>bFrameThumb</i>	Show a frame around thumb?
in	<i>colFrame</i>	Color of thumb frame
in	<i>nTickDiv</i>	Number of tick divisions to show (0 for none)
in	<i>nTickLen</i>	Length of tick marks
in	<i>colTick</i>	Color of ticks

## Returns

none

## 9.29.3.7 gslc\_ElemXSeekBarTouch()

```
bool gslc_ElemXSeekBarTouch (
    void * pvGui,
```

```

void * pvElemRef,
gslc_teTouch eTouch,
int16_t nRelX,
int16_t nRelY )

```

Handle touch events to Seekbar element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

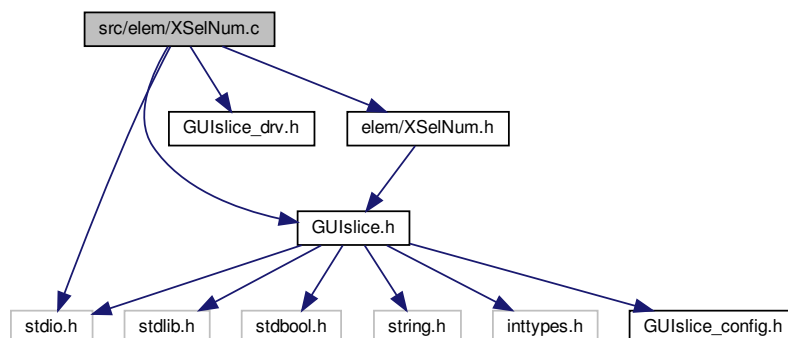
## 9.30 src/elem/XSelNum.c File Reference

```

#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XSelNum.h"
#include <stdio.h>

```

Include dependency graph for XSelNum.c:



## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXSelNumCreate](#) ([gslc\\_tsGui](#) \*pGui, `int16_t` nElemId, `int16_t` nPage, [gslc\\_tsXSelNum](#) \*pXData, [gslc\\_tsRect](#) rElem, `int8_t` nFontId)

*Create a SelNum Element.*

- bool [gslc\\_ElemXSelNumDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw a SelNum element on the screen.*

- int [gslc\\_ElemXSelNumGetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSelNum](#) \*pSelNum)

*Get the current counter associated with SelNum.*

- void [gslc\\_ElemXSelNumSetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSelNum](#) \*pSelNum, int16\_t nCount)

*Set the current counter associated with SelNum.*

- bool [gslc\\_ElemXSelNumClick](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nX, int16\_t nY)

*Handle a click event within the SelNum.*

- bool [gslc\\_ElemXSelNumTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)

*Handle touch (up,down,move) events to SelNum element.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []
- static const int16\_t [SELNUM\\_ID\\_BTN\\_INC](#)
- static const int16\_t [SELNUM\\_ID\\_BTN\\_DEC](#)
- static const int16\_t [SELNUM\\_ID\\_TXT](#)

## 9.30.1 Function Documentation

### 9.30.1.1 [gslc\\_ElemXSelNumClick\(\)](#)

```
bool gslc_ElemXSelNumClick (
    void * pvGui,
    void * pvElemRef,
    gslc\_teTouch eTouch,
    int16_t nX,
    int16_t nY )
```

Handle a click event within the SelNum.

- This is called internally by the SelNum touch handler

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <a href="#">gslc_tsGui*</a> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <a href="#">gslc_tsElemRef*</a> )
in	<i>eTouch</i>	Touch event type
in	<i>nX</i>	Touch X coord
in	<i>nY</i>	Touch Y coord

**Returns**

none

**9.30.1.2 gslc\_ElemXSelNumCreate()**

```

gslc_tsElemRef* gslc_ElemXSelNumCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSelNum * pXData,
    gslc_tsRect rElem,
    int8_t nFontId )

```

Create a SelNum Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>nFontId</i>	Font ID to use for drawing the element

**Returns**

Pointer to Element or NULL if failure

**9.30.1.3 gslc\_ElemXSelNumDraw()**

```

bool gslc_ElemXSelNumDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a SelNum element on the screen.

- Called during redraw

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.30.1.4 gslc\_ElemXSelNumGetCounter()**

```
int gslc_ElemXSelNumGetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSelNum * pSelNum )
```

Get the current counter associated with SelNum.

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pSelNum</i>	Ptr to Element

**Returns**

Current counter value

**9.30.1.5 gslc\_ElemXSelNumSetCounter()**

```
void gslc_ElemXSelNumSetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSelNum * pSelNum,
    int16_t nCount )
```

Set the current counter associated with SelNum.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pSelNum</i>	Ptr to Element
in	<i>nCount</i>	New counter value

**Returns**

none

**9.30.1.6 gslc\_ElemXSelNumTouch()**

```
bool gslc_ElemXSelNumTouch (
    void * pvGui,
```

```

void * pvElemRef,
gslc_teTouch eTouch,
int16_t nRelX,
int16_t nRelY )

```

Handle touch (up,down,move) events to SelNum element.

- Called from `gslc_ElemSendEventTouch()`

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

## 9.30.2 Variable Documentation

### 9.30.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

### 9.30.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

### 9.30.2.3 SELNUM\_ID\_BTN\_DEC

```
const int16_t SELNUM_ID_BTN_DEC [static]
```

### 9.30.2.4 SELNUM\_ID\_BTN\_INC

```
const int16_t SELNUM_ID_BTN_INC [static]
```



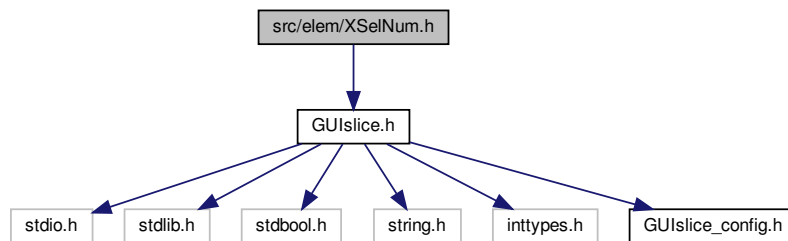
## 9.30.2.5 SELNUM\_ID\_TXT

```
const int16_t SELNUM_ID_TXT [static]
```

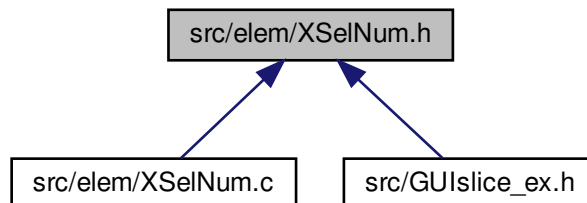
## 9.31 src/elem/XSelNum.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XSelNum.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXSelNum](#)  
*Extended data for SelNum element.*

## Macros

- #define [GSLC\\_TYPEX\\_SELNUM](#)
- #define [SELNUM\\_STR\\_LEN](#)

## Functions

- `gslc_tsElemRef * gslc_ElemXSelNumCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsXSelNum *pXData, gslc_tsRect rElem, int8_t nFontId)`  
*Create a SelNum Element.*
- `bool gslc_ElemXSelNumDraw (void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`  
*Draw a SelNum element on the screen.*
- `int gslc_ElemXSelNumGetCounter (gslc_tsGui *pGui, gslc_tsXSelNum *pSelNum)`  
*Get the current counter associated with SelNum.*
- `void gslc_ElemXSelNumSetCounter (gslc_tsGui *pGui, gslc_tsXSelNum *pSelNum, int16_t nCount)`  
*Set the current counter associated with SelNum.*
- `bool gslc_ElemXSelNumClick (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nX, int16_t nY)`  
*Handle a click event within the SelNum.*
- `bool gslc_ElemXSelNumTouch (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
*Handle touch (up,down,move) events to SelNum element.*

### 9.31.1 Macro Definition Documentation

#### 9.31.1.1 GSLC\_TYPEX\_SELNUM

```
#define GSLC_TYPEX_SELNUM
```

#### 9.31.1.2 SELNUM\_STR\_LEN

```
#define SELNUM_STR_LEN
```

### 9.31.2 Function Documentation

#### 9.31.2.1 gslc\_ElemXSelNumClick()

```
bool gslc_ElemXSelNumClick (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nX,
    int16_t nY )
```

Handle a click event within the SelNum.

- This is called internally by the SelNum touch handler

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nX</i>	Touch X coord
in	<i>nY</i>	Touch Y coord

## Returns

none

## 9.31.2.2 gslc\_ElemXSelNumCreate()

```
gslc_tsElemRef* gslc_ElemXSelNumCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSelNum * pXData,
    gslc_tsRect rElem,
    int8_t nFontId )
```

Create a SelNum Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>nFontId</i>	Font ID to use for drawing the element

## Returns

Pointer to Element or NULL if failure

## 9.31.2.3 gslc\_ElemXSelNumDraw()

```
bool gslc_ElemXSelNumDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a SelNum element on the screen.

- Called during redraw

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.31.2.4 gslc\_ElemXSelNumGetCounter()**

```
int gslc_ElemXSelNumGetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSelNum * pSelNum )
```

Get the current counter associated with SelNum.

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pSelNum</i>	Ptr to Element

**Returns**

Current counter value

**9.31.2.5 gslc\_ElemXSelNumSetCounter()**

```
void gslc_ElemXSelNumSetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSelNum * pSelNum,
    int16_t nCount )
```

Set the current counter associated with SelNum.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pSelNum</i>	Ptr to Element
in	<i>nCount</i>	New counter value

**Returns**

none

**9.31.2.6 gslc\_ElemXSelNumTouch()**

```
bool gslc_ElemXSelNumTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch (up,down,move) events to SelNum element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

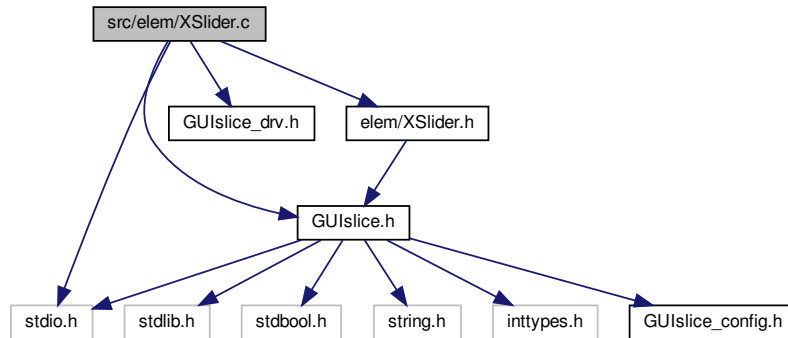
**Returns**

true if success, false otherwise

**9.32 src/elem/XSlider.c File Reference**

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XSlider.h"
#include <stdio.h>
```

Include dependency graph for XSlider.c:



## Functions

- `gslc_tsElemRef * gslc_ElemXSliderCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXSlider *pXData, gslc\_tsRect rElem, int16_t nPosMin, int16_t nPosMax, int16_t nPos, uint16_t nThumbSz, bool bVert)`  
*Create a Slider Element.*
- `void gslc\_ElemXSliderSetStyle (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bTrim, gslc\_tsColor colTrim, uint16_t nTickDiv, int16_t nTickLen, gslc\_tsColor colTick)`  
*Set a Slider element's current position.*
- `void gslc\_ElemXSliderSetSnapEn (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bSnapEn)`  
*Enable touch to snap to the nearest tick mark.*
- `int gslc\_ElemXSliderGetPos (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
*Get a Slider element's current position.*
- `void gslc\_ElemXSliderSetPos (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, int16_t nPos)`  
*Set a Slider element's current position.*
- `void gslc\_ElemXSliderSetPosFunc (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, GSLC\_CB\_XSLIDER\_POS funcCb)`  
*Assign the position callback function for a slider.*
- `bool gslc\_ElemXSliderDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
*Draw a Slider element on the screen.*
- `bool gslc\_ElemXSliderTouch (void *pvGui, void *pvElemRef, gslc\_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
*Handle touch events to Slider element.*

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

### 9.32.1 Function Documentation

## 9.32.1.1 gslc\_ElemXSliderCreate()

```

gslc_tsElemRef* gslc_ElemXSliderCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSlider * pXData,
    gslc_tsRect rElem,
    int16_t nPosMin,
    int16_t nPosMax,
    int16_t nPos,
    uint16_t nThumbSz,
    bool bVert )

```

Create a Slider Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nPosMin</i>	Minimum position value
in	<i>nPosMax</i>	Maximum position value
in	<i>nPos</i>	Starting position value
in	<i>nThumbSz</i>	Size of the thumb control
in	<i>bVert</i>	Orientation (true for vertical)

## Returns

Pointer to Element reference or NULL if failure

## 9.32.1.2 gslc\_ElemXSliderDraw()

```

bool gslc_ElemXSliderDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a Slider element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.32.1.3 gslc\_ElemXSliderGetPos()**

```
int gslc_ElemXSliderGetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Slider element's current position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Current slider position

**9.32.1.4 gslc\_ElemXSliderSetPos()**

```
void gslc_ElemXSliderSetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nPos )
```

Set a Slider element's current position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nPos</i>	New position value

**Returns**

none



## 9.32.1.5 gslc\_ElemXSliderSetPosFunc()

```
void gslc_ElemXSliderSetPosFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XSLIDER_POS funcCb )
```

Assign the position callback function for a slider.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to position routine (or NULL for none)

## Returns

none

## 9.32.1.6 gslc\_ElemXSliderSetSnapEn()

```
void gslc_ElemXSliderSetSnapEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bSnapEn )
```

Enable touch to snap to the nearest tick mark.

- nTickDiv (in SetStyle) must be non-zero for snap to be active

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bSnapEn</i>	Enable snap function if true

## Returns

none

## 9.32.1.7 gslc\_ElemXSliderSetStyle()

```
void gslc_ElemXSliderSetStyle (
    gslc_tsGui * pGui,
```

```

gslc_tsElemRef * pElemRef,
bool bTrim,
gslc_tsColor colTrim,
uint16_t nTickDiv,
int16_t nTickLen,
gslc_tsColor colTick )

```

Set a Slider element's current position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bTrim</i>	Show a colored trim?
in	<i>colTrim</i>	Color of trim
in	<i>nTickDiv</i>	Number of tick divisions to show (0 for none)
in	<i>nTickLen</i>	Length of tickmarks
in	<i>colTick</i>	Color of ticks

#### Returns

none

#### 9.32.1.8 gslc\_ElemXSliderTouch()

```

bool gslc_ElemXSliderTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )

```

Handle touch events to Slider element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

### 9.32.2 Variable Documentation

#### 9.32.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

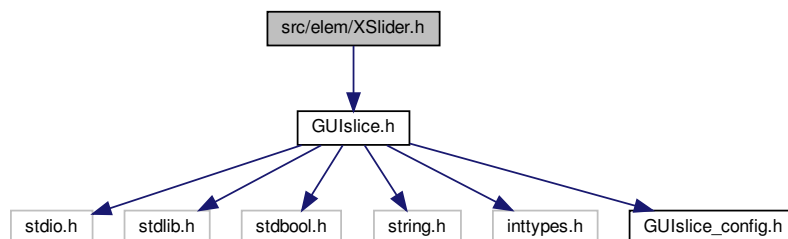
#### 9.32.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

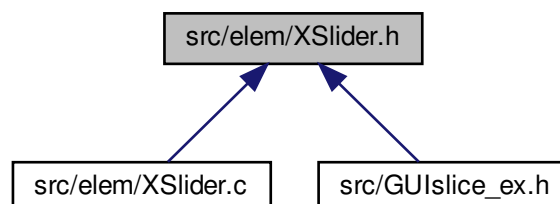
## 9.33 src/elem/XSlider.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XSlider.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXSlider](#)  
*Extended data for Slider element.*

## Macros

- #define [GSLC\\_TYPEX\\_SLIDER](#)
- #define [gslc\\_ElemXSliderCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, nPosMin\_, nPosMax\_, nPos\_, nThumbSz\_, bVert\_, colFrame\_, colFill\_)  
*Create a Slider Element in Flash.*

## Typedefs

- typedef bool(\* [GSLC\\_CB\\_XSLIDER\\_POS](#)) (void \*pvGui, void \*pvElem, int16\_t nPos)  
*Callback function for slider feedback.*

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXSliderCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXSlider](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nPosMin, int16\_t nPosMax, int16\_t nPos, uint16\_t nThumbSz, bool bVert)  
*Create a Slider Element.*
- void [gslc\\_ElemXSliderSetStyle](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bTrim, [gslc\\_tsColor](#) colTrim, uint16\_t nTickDiv, int16\_t nTickLen, [gslc\\_tsColor](#) colTick)  
*Set a Slider element's current position.*
- void [gslc\\_ElemXSliderSetSnapEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bSnapEn)  
*Enable touch to snap to the nearest tick mark.*
- int [gslc\\_ElemXSliderGetPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a Slider element's current position.*
- void [gslc\\_ElemXSliderSetPos](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nPos)  
*Set a Slider element's current position.*
- void [gslc\\_ElemXSliderSetPosFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_XSLIDER\\_POS](#) funcCb)  
*Assign the position callback function for a slider.*
- bool [gslc\\_ElemXSliderDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Slider element on the screen.*
- bool [gslc\\_ElemXSliderTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)  
*Handle touch events to Slider element.*

### 9.33.1 Macro Definition Documentation

## 9.33.1.1 gslc\_ElemXSliderCreate\_P

```
#define gslc_ElemXSliderCreate_P(
    pGui,
    nElemId,
    nPage,
    nX,
    nY,
    nW,
    nH,
    nPosMin_,
    nPosMax_,
    nPos_,
    nThumbSz_,
    bVert_,
    colFrame_,
    colFill_ )
```

Create a Slider Element in Flash.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Unique element ID to assign
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>nPosMin</i> ↔ _	Minimum position value
in	<i>nPosMax</i> ↔ _	Maximum position value
in	<i>nPos</i> _	Starting position value
in	<i>nThumb</i> ↔ <i>Sz</i> _	Size of the thumb control
in	<i>bVert</i> _	Orientation (true for vertical)
in	<i>colFrame</i> ↔ _	Color of the element frame
in	<i>colFill</i> _	Color of the element fill

## Returns

none

## 9.33.1.2 GSLC\_TYPEX\_SLIDER

```
#define GSLC_TYPEX_SLIDER
```

### 9.33.2 Typedef Documentation

#### 9.33.2.1 GSLC\_CB\_XSLIDER\_POS

```
typedef bool(* GSLC_CB_XSLIDER_POS) (void *pvGui, void *pvElem, int16_t nPos)
```

Callback function for slider feedback.

### 9.33.3 Function Documentation

#### 9.33.3.1 gslc\_ElemXSliderCreate()

```
gslc_tsElemRef* gslc_ElemXSliderCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSlider * pXData,
    gslc_tsRect rElem,
    int16_t nPosMin,
    int16_t nPosMax,
    int16_t nPos,
    uint16_t nThumbSz,
    bool bVert )
```

Create a Slider Element.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining checkbox size
in	<i>nPosMin</i>	Minimum position value
in	<i>nPosMax</i>	Maximum position value
in	<i>nPos</i>	Starting position value
in	<i>nThumbSz</i>	Size of the thumb control
in	<i>bVert</i>	Orientation (true for vertical)

##### Returns

Pointer to Element reference or NULL if failure

9.33.3.2 `gslc_ElemXSliderDraw()`

```
bool gslc_ElemXSliderDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Slider element on the screen.

- Called from `gslc_ElemDraw()`

## Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.33.3.3 `gslc_ElemXSliderGetPos()`

```
int gslc_ElemXSliderGetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Slider element's current position.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

Current slider position

9.33.3.4 `gslc_ElemXSliderSetPos()`

```
void gslc_ElemXSliderSetPos (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    int16_t nPos )
```

Set a Slider element's current position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nPos</i>	New position value

**Returns**

none

**9.33.3.5 gslc\_ElemXSliderSetPosFunc()**

```
void gslc_ElemXSliderSetPosFunc (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    GSLC_CB_XSLIDER_POS funcCb )
```

Assign the position callback function for a slider.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>funcCb</i>	Function pointer to position routine (or NULL for none)

**Returns**

none

**9.33.3.6 gslc\_ElemXSliderSetSnapEn()**

```
void gslc_ElemXSliderSetSnapEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bSnapEn )
```

Enable touch to snap to the nearest tick mark.

- nTickDiv (in SetStyle) must be non-zero for snap to be active



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bSnapEn</i>	Enable snap function if true

## Returns

none

## 9.33.3.7 gslc\_ElemXSliderSetStyle()

```
void gslc_ElemXSliderSetStyle (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bTrim,
    gslc_tsColor colTrim,
    uint16_t nTickDiv,
    int16_t nTickLen,
    gslc_tsColor colTick )
```

Set a Slider element's current position.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bTrim</i>	Show a colored trim?
in	<i>colTrim</i>	Color of trim
in	<i>nTickDiv</i>	Number of tick divisions to show (0 for none)
in	<i>nTickLen</i>	Length of tickmarks
in	<i>colTick</i>	Color of ticks

## Returns

none

## 9.33.3.8 gslc\_ElemXSliderTouch()

```
bool gslc_ElemXSliderTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Slider element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

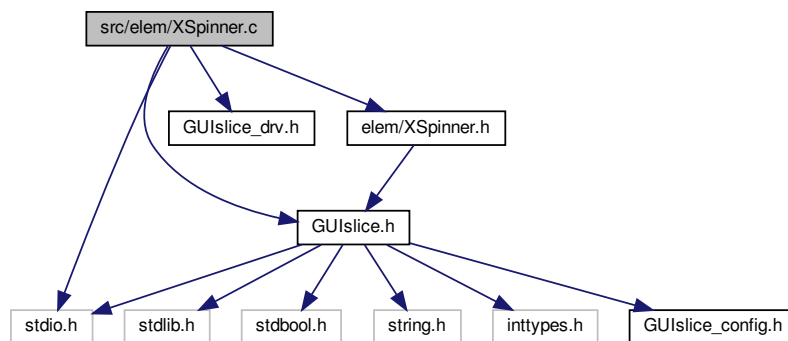
#### Returns

true if success, false otherwise

## 9.34 src/elem/XSpinner.c File Reference

Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound.

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XSpinner.h"
#include <stdio.h>
Include dependency graph for XSpinner.c:
```



#### Functions

- `gslc_tsElemRef * gslc_ElemXSpinnerCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXSpinner *pXData, gslc\_tsRect rElem, int16_t nMin, int16_t nMax, int16_t nVal, int16_t nIncr, int8_t nFontId, int8_t nButtonSz, GSLC\_CB\_INPUT cbInput)`  
Create a Spinner Element.
- `bool gslc\_ElemXSpinnerSetChars (void *pvGui, gslc\_tsElemRef *pElemRef, uint8_t cIncr, uint8_t cDecr)`  
Set Up and Down characters for the Spinner element.
- `void gslc\_XSpinnerDrawVirtualTxt (gslc_tsGui *pGui, gslc\_tsRect rElem, int16_t nFontId, char *pValStr, int8_t eTxtAlign, gslc\_tsColor cColFrame, gslc\_tsColor cColFill, gslc\_tsColor cColTxt)`

- void [gslc\\_XSpinnerDrawVirtualBtn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBuf↵Max, int16\_t nFontId, [gslc\\_tsColor](#) cColFrame, [gslc\\_tsColor](#) cColFill, [gslc\\_tsColor](#) cColFillGlow, [gslc\\_tsColor](#) cColText, bool bRoundedEn, bool bGlow, bool bFocus)
- bool [gslc\\_ElemXSpinnerDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Spinner element on the screen.*
- int [gslc\\_ElemXSpinnerGetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSpinner](#) \*pSpinner)  
*Get the current counter associated with Spinner.*
- void [gslc\\_ElemXSpinnerSetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSpinner](#) \*pSpinner, int16\_t nCount)  
*Set the current counter associated with Spinner.*
- int16\_t [gslc\\_XSpinnerMapEvent](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSpinner](#) \*pConfig, int16\_t nRelX, int16\_t nRelY)
- bool [gslc\\_ElemXSpinnerTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)  
*Handle touch (up,down,move) events to Spinner element.*

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []
- static const int16\_t [SPINNER\\_ID\\_BTN\\_INC](#)
- static const int16\_t [SPINNER\\_ID\\_BTN\\_DEC](#)
- static const int16\_t [SPINNER\\_ID\\_TXT](#)

## 9.34.1 Detailed Description

Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound.

## 9.34.2 Function Documentation

### 9.34.2.1 [gslc\\_ElemXSpinnerCreate\(\)](#)

```
gslc\_tsElemRef* gslc\_ElemXSpinnerCreate (
    gslc\_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc\_tsXSpinner * pXData,
    gslc\_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    int16_t nIncr,
    int8_t nFontId,
    int8_t nButtonSz,
    GSLC\_CB\_INPUT cbInput )
```

Create a Spinner Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining overall size
in	<i>nMin</i>	Minimum value of Spinner
in	<i>nMax</i>	Maximum value of Spinner
in	<i>nVal</i>	Starting value of Spinner
in	<i>nIncr</i>	Increment Spinner by this value
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>nButtonSz</i>	Size of individual buttons
in	<i>cbInput</i>	Callback for touch events

**Returns**

Pointer to Element or NULL if failure

**9.34.2.2 gslc\_ElemXSpinnerDraw()**

```
bool gslc_ElemXSpinnerDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Spinner element on the screen.

- Called during redraw

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.34.2.3 gslc\_ElemXSpinnerGetCounter()**

```
int gslc_ElemXSpinnerGetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSpinner * pSpinner )
```

Get the current counter associated with Spinner.

#### Parameters

in	<i>pGui</i>	Ptr to GUI
in	<i>pSpinner</i>	Ptr to Element

#### Returns

Current counter value

#### 9.34.2.4 gslc\_ElemXSpinnerSetChars()

```
bool gslc_ElemXSpinnerSetChars (
    void * pvGui,
    gslc_tsElemRef * pElemRef,
    uint8_t cIncr,
    uint8_t cDecr )
```

Set Up and Down characters for the Spinner element.

- Called during redraw

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pElemRef</i>	Ptr to ElementRef
in	<i>cIncr</i>	Character to use to indicate incrementing the spinner
in	<i>cDecr</i>	Character to use to indicate decrementing the spinner

#### Returns

true if success, false otherwise

#### 9.34.2.5 gslc\_ElemXSpinnerSetCounter()

```
void gslc_ElemXSpinnerSetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSpinner * pSpinner,
    int16_t nCount )
```

Set the current counter associated with Spinner.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pSpinner</i>	Ptr to Element
in	<i>nCount</i>	New counter value

## Returns

none

9.34.2.6 `gslc_ElemXSpinnerTouch()`

```
bool gslc_ElemXSpinnerTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch (up,down,move) events to Spinner element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

## Returns

true if success, false otherwise

9.34.2.7 `gslc_XSpinnerDrawVirtualBtn()`

```
void gslc_XSpinnerDrawVirtualBtn (
    gslc_tsGui * pGui,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId,
```

```
gslc_tsColor cColFrame,  
gslc_tsColor cColFill,  
gslc_tsColor cColFillGlow,  
gslc_tsColor cColText,  
bool bRoundedEn,  
bool bGlow,  
bool bFocus )
```

#### 9.34.2.8 gslc\_XSpinnerDrawVirtualTxt()

```
void gslc_XSpinnerDrawVirtualTxt (  
    gslc_tsGui * pGui,  
    gslc_tsRect rElem,  
    int16_t nFontId,  
    char * pValStr,  
    int8_t eTxtAlign,  
    gslc_tsColor cColFrame,  
    gslc_tsColor cColFill,  
    gslc_tsColor cColTxt )
```

#### 9.34.2.9 gslc\_XSpinnerMapEvent()

```
int16_t gslc_XSpinnerMapEvent (  
    gslc_tsGui * pGui,  
    gslc_tsXSpinner * pConfig,  
    int16_t nRelX,  
    int16_t nRelY )
```

### 9.34.3 Variable Documentation

#### 9.34.3.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

#### 9.34.3.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

### 9.34.3.3 SPINNER\_ID\_BTN\_DEC

```
const int16_t SPINNER_ID_BTN_DEC [static]
```

### 9.34.3.4 SPINNER\_ID\_BTN\_INC

```
const int16_t SPINNER_ID_BTN_INC [static]
```

### 9.34.3.5 SPINNER\_ID\_TXT

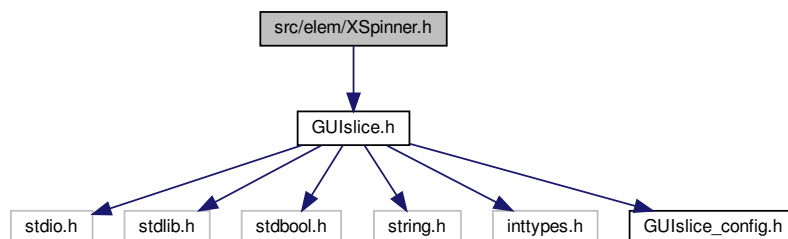
```
const int16_t SPINNER_ID_TXT [static]
```

## 9.35 src/elem/XSpinner.h File Reference

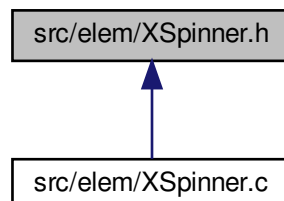
Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound.

```
#include "GUIslice.h"
```

Include dependency graph for XSpinner.h:



This graph shows which files directly or indirectly include this file:





## Data Structures

- struct [gslc\\_tsXSpinner](#)  
*Extended data for Spinner element.*

## Macros

- #define [GSLC\\_TYPEX\\_SPINNER](#)
- #define [XSPINNER\\_COMP\\_CNT](#)
- #define [XSPINNER\\_STR\\_LEN](#)
- #define [XSPINNER\\_CB\\_STATE\\_UPDATE](#)

## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXSpinnerCreate](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nElemId, [int16\\_t](#) nPage, [gslc\\_tsXSpinner](#) \*pXData, [gslc\\_tsRect](#) rElem, [int16\\_t](#) nMin, [int16\\_t](#) nMax, [int16\\_t](#) nVal, [int16\\_t](#) nIncr, [int8\\_t](#) nFontId, [int8\\_t](#) nButtonSz, [GSLC\\_CB\\_INPUT](#) cbInput)  
*Create a Spinner Element.*
- [bool gslc\\_ElemXSpinnerSetChars](#) ([void](#) \*pvGui, [gslc\\_tsElemRef](#) \*pElemRef, [uint8\\_t](#) cIncr, [uint8\\_t](#) cDecr)  
*Set Up and Down characters for the Spinner element.*
- [bool gslc\\_ElemXSpinnerDraw](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Spinner element on the screen.*
- [int gslc\\_ElemXSpinnerGetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSpinner](#) \*pSpinner)  
*Get the current counter associated with Spinner.*
- [void gslc\\_ElemXSpinnerSetCounter](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsXSpinner](#) \*pSpinner, [int16\\_t](#) nCount)  
*Set the current counter associated with Spinner.*
- [bool gslc\\_ElemXSpinnerClick](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teTouch](#) eTouch, [int16\\_t](#) nX, [int16\\_t](#) nY)  
*Handle a click event within the Spinner.*
- [bool gslc\\_ElemXSpinnerTouch](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teTouch](#) eTouch, [int16\\_t](#) nRelX, [int16\\_t](#) nRelY)  
*Handle touch (up,down,move) events to Spinner element.*

### 9.35.1 Detailed Description

Author: Paul Conti Date: 2022-04-06 Modified to use Virtual Elements instead of Compound.

### 9.35.2 Macro Definition Documentation

#### 9.35.2.1 GSLC\_TYPEX\_SPINNER

```
#define GSLC_TYPEX_SPINNER
```

### 9.35.2.2 XSPINNER\_CB\_STATE\_UPDATE

```
#define XSPINNER_CB_STATE_UPDATE
```

### 9.35.2.3 XSPINNER\_COMP\_CNT

```
#define XSPINNER_COMP_CNT
```

### 9.35.2.4 XSPINNER\_STR\_LEN

```
#define XSPINNER_STR_LEN
```

## 9.35.3 Function Documentation

### 9.35.3.1 gslc\_ElemXSpinnerClick()

```
bool gslc_ElemXSpinnerClick (
    void * pvGui,
    void * pvElemRef,
    gslc\_teTouch eTouch,
    int16_t nX,
    int16_t nY )
```

Handle a click event within the Spinner.

- This is called internally by the Spinner touch handler

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nX</i>	Touch X coord
in	<i>nY</i>	Touch Y coord

#### Returns

none

## 9.35.3.2 gslc\_ElemXSpinnerCreate()

```

gslc_tsElemRef* gslc_ElemXSpinnerCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXSpinner * pXData,
    gslc_tsRect rElem,
    int16_t nMin,
    int16_t nMax,
    int16_t nVal,
    int16_t nIncr,
    int8_t nFontId,
    int8_t nButtonSz,
    GSLC_CB_INPUT cbInput )

```

Create a Spinner Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining overall size
in	<i>nMin</i>	Minimum value of Spinner
in	<i>nMax</i>	Maximum value of Spinner
in	<i>nVal</i>	Starting value of Spinner
in	<i>nIncr</i>	Increment Spinner by this value
in	<i>nFontId</i>	Font ID to use for drawing the element
in	<i>nButtonSz</i>	Size of individual buttons
in	<i>cbInput</i>	Callback for touch events

## Returns

Pointer to Element or NULL if failure

## 9.35.3.3 gslc\_ElemXSpinnerDraw()

```

bool gslc_ElemXSpinnerDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a Spinner element on the screen.

- Called during redraw

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.35.3.4 `gslc_ElemXSpinnerGetCounter()`**

```
int gslc_ElemXSpinnerGetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSpinner * pSpinner )
```

Get the current counter associated with Spinner.

**Parameters**

in	<i>pGui</i>	Ptr to GUI
in	<i>pSpinner</i>	Ptr to Element

**Returns**

Current counter value

**9.35.3.5 `gslc_ElemXSpinnerSetChars()`**

```
bool gslc_ElemXSpinnerSetChars (
    void * pvGui,
    gslc_tsElemRef * pElemRef,
    uint8_t cIncr,
    uint8_t cDecr )
```

Set Up and Down characters for the Spinner element.

- Called during redraw

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pElemRef</i>	Ptr to ElementRef
in	<i>cIncr</i>	Character to use to indicate incrementing the spinner
in	<i>cDecr</i>	Character to use to indicate decrementing the spinner

**Returns**

true if success, false otherwise

**9.35.3.6 gslc\_ElemXSpinnerSetCounter()**

```
void gslc_ElemXSpinnerSetCounter (
    gslc_tsGui * pGui,
    gslc_tsXSpinner * pSpinner,
    int16_t nCount )
```

Set the current counter associated with Spinner.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pSpinner</i>	Ptr to Element
in	<i>nCount</i>	New counter value

**Returns**

none

**9.35.3.7 gslc\_ElemXSpinnerTouch()**

```
bool gslc_ElemXSpinnerTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch (up,down,move) events to Spinner element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

**Parameters**

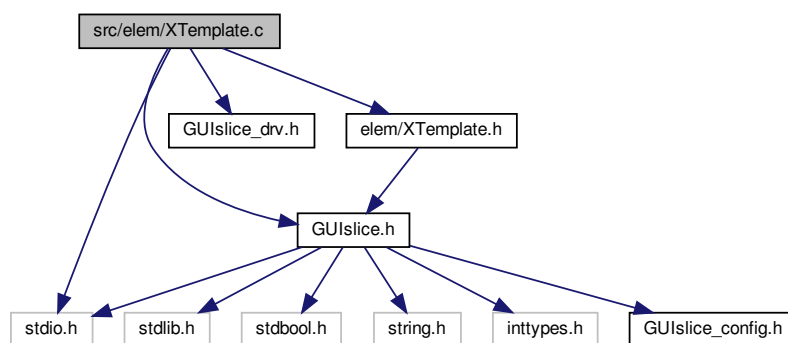
in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

## Returns

true if success, false otherwise

## 9.36 src/elem/XTemplate.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XTemplate.h"
#include <stdio.h>
Include dependency graph for XTemplate.c:
```



## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXTemplateCreate](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nElemId, [int16\\_t](#) nPage, [gslc\\_tsXTemplate](#) \*pXData, [gslc\\_tsRect](#) rElem, [char](#) \*pStrBuf, [uint8\\_t](#) nStrBufMax, [int16\\_t](#) nFontId)  
*Create an Extended Text Field Element.*
- [bool gslc\\_ElemXTemplateDraw](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw the template element on the screen.*
- [bool gslc\\_ElemXTemplateTouch](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teTouch](#) eTouch, [int16\\_t](#) nRelX, [int16\\_t](#) nRelY)  
*Handle touch events to template element.*

## Variables

- [const char](#) [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- [const char](#) [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

### 9.36.1 Function Documentation

9.36.1.1 `gslc_ElemXTemplateCreate()`

```

gslc_tsElemRef* gslc_ElemXTemplateCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTemplate * pXData,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )

```

Create an Extended Text Field Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pStrBuf</i>	Ptr to string buffer
in	<i>nStrBufMax</i>	Maximum buffer alength allocated to pStrBuf
in	<i>nFontId</i>	ID of font to use for text output

## Returns

Pointer to Element reference or NULL if failure

9.36.1.2 `gslc_ElemXTemplateDraw()`

```

bool gslc_ElemXTemplateDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw the template element on the screen.

- Called from `gslc_ElemDraw()`

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.36.1.3 gslc\_ElemXTemplateTouch()**

```
bool gslc_ElemXTemplateTouch (
    void * pvGui,
    void * pvElemRef,
    gslc\_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to template element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

**Returns**

true if success, false otherwise

**9.36.2 Variable Documentation****9.36.2.1 ERRSTR\_NULL**

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

**9.36.2.2 ERRSTR\_PXD\_NULL**

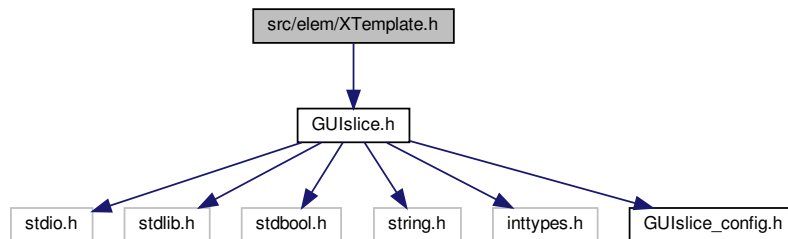
```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```



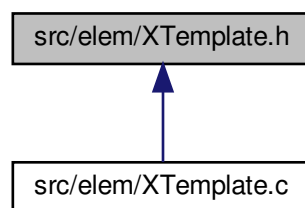
## 9.37 src/elem/XTemplate.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XTemplate.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gslc\\_tsXTemplate](#)  
*Callback function for slider feedback.*

### Macros

- `#define` [GSLC\\_TYPEX\\_TEMPLATE](#)

### Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXTemplateCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXTemplate](#) \*pXData, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create an Extended Text Field Element.*
- bool [gslc\\_ElemXTemplateDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw the template element on the screen.*
- bool [gslc\\_ElemXTemplateTouch](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nRelX, int16\_t nRelY)  
*Handle touch events to template element.*

## 9.37.1 Macro Definition Documentation

### 9.37.1.1 GSLC\_TYPEX\_TEMPLATE

```
#define GSLC_TYPEX_TEMPLATE
```

## 9.37.2 Function Documentation

### 9.37.2.1 gslc\_ElemXTemplateCreate()

```
gslc_tsElemRef* gslc_ElemXTemplateCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTemplate * pXData,
    gslc_tsRect rElem,
    char * pStrBuf,
    uint8_t nStrBufMax,
    int16_t nFontId )
```

Create an Extended Text Field Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining element size
in	<i>pStrBuf</i>	Ptr to string buffer
in	<i>nStrBufMax</i>	Maximum buffer alength allocated to pStrBuf
in	<i>nFontId</i>	ID of font to use for text output

#### Returns

Pointer to Element reference or NULL if failure

### 9.37.2.2 gslc\_ElemXTemplateDraw()

```
bool gslc_ElemXTemplateDraw (
    void * pvGui,
```

```
void * pvElemRef,
gslc_teRedrawType eRedraw )
```

Draw the template element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

#### 9.37.2.3 [gslc\\_ElemXTemplateTouch\(\)](#)

```
bool gslc_ElemXTemplateTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to template element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

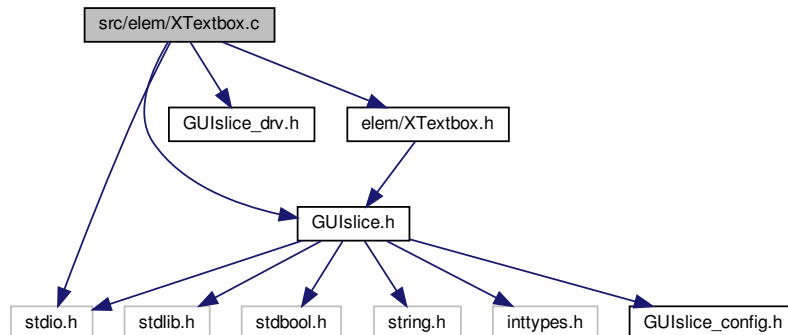
#### Returns

true if success, false otherwise

## 9.38 src/elem/XTextbox.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
```

```
#include "elem/XTextbox.h"
#include <stdio.h>
Include dependency graph for XTextbox.c:
```



## Functions

- `gslc_tsElemRef * gslc_ElemXTextboxCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXTextbox *pXData, gslc\_tsRect rElem, int16_t nFontId, char *pBuf, uint16_t nBufRows, uint16_t nBufCols)`  
*Create a Textbox Element.*
- `void gslc_ElemXTextboxReset (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
*Reset the contents of the textbox.*
- `void gslc_ElemXTextboxLineWrAdv (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`
- `void gslc_ElemXTextboxScrollSet (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)`  
*Set the textbox scroll position (nScrollPos) as a fraction of nScrollMax.*
- `void gslc_ElemXTextboxBufAdd (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, unsigned char chNew, bool bAdvance)`
- `void gslc_ElemXTextboxColSet (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsColor nCol)`  
*Insert a color set code into the current buffer position.*
- `void gslc_ElemXTextboxColReset (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
*Insert a color reset code into the current buffer position.*
- `void gslc_ElemXTextboxWrapSet (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bWrapEn)`  
*Enable or disable line wrap within textbox.*
- `void gslc_ElemXTextboxAdd (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, char *pTxt)`  
*Add a text string to the textbox.*
- `bool gslc_ElemXTextboxDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
*Draw a Textbox element on the screen.*

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

### 9.38.1 Function Documentation

### 9.38.1.1 gslc\_ElemXTextboxAdd()

```
void gslc_ElemXTextboxAdd (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    char * pTxt )
```

Add a text string to the textbox.

- If it includes a newline then the buffer will advance to the next row
- If wrap has been enabled, then a newline will be forced

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>pTxt</i>	Pointer to text string (null-terminated)

#### Returns

none

### 9.38.1.2 gslc\_ElemXTextboxBufAdd()

```
void gslc_ElemXTextboxBufAdd (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    unsigned char chNew,
    bool bAdvance )
```

### 9.38.1.3 gslc\_ElemXTextboxColReset()

```
void gslc_ElemXTextboxColReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Insert a color reset code into the current buffer position.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

none

**9.38.1.4 gslc\_ElemXTextboxColSet()**

```
void gslc_ElemXTextboxColSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor nCol )
```

Insert a color set code into the current buffer position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nCol</i>	Color to assign for next text written to textbox

**Returns**

none

**9.38.1.5 gslc\_ElemXTextboxCreate()**

```
gslc_tsElemRef* gslc_ElemXTextboxCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTextbox * pXData,
    gslc_tsRect rElem,
    int16_t nFontId,
    char * pBuf,
    uint16_t nBufRows,
    uint16_t nBufCols )
```

Create a Textbox Element.

- The textbox is a scrolling window designed for displaying multi-line text using a monospaced font. A character buffer is defined by nBufRows\*nBufCols to capture the added text. If the allocation buffer is larger than the display size (defined by rElem), then a scrollbar will be shown.
- Support for changing color within a row can be enabled with `GSLC_FEATURE_XTEXTBOX_EMBED 1`
- Note that each color change command will consume 4 of the available "column" bytes.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining textbox size
in	<i>nFontId</i>	Font ID to use for text area
in	<i>pBuf</i>	Ptr to text buffer (already allocated) with size (nBufRows*nBufCols) chars
in	<i>nBufRows</i>	Number of rows in buffer
in	<i>nBufCols</i>	Number of columns in buffer (incl special codes)

## Returns

Pointer to Element reference or NULL if failure

9.38.1.6 `gslc_ElemXTextboxDraw()`

```
bool gslc_ElemXTextboxDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Textbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

9.38.1.7 `gslc_ElemXTextboxLineWrAdv()`

```
void gslc_ElemXTextboxLineWrAdv (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

### 9.38.1.8 gslc\_ElemXTextboxReset()

```
void gslc_ElemXTextboxReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Reset the contents of the textbox.

- Clears the buffer and resets the position

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

none

### 9.38.1.9 gslc\_ElemXTextboxScrollSet()

```
void gslc_ElemXTextboxScrollSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nScrollPos,
    uint8_t nScrollMax )
```

Set the textbox scroll position (nScrollPos) as a fraction of nScrollMax.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	New scroll position
in	<i>nScrollMax</i>	Maximum scroll position

#### Returns

none

### 9.38.1.10 gslc\_ElemXTextboxWrapSet()

```
void gslc_ElemXTextboxWrapSet (
    gslc_tsGui * pGui,
```



```
gslc_tsElemRef * pElemRef,
bool bWrapEn )
```

Enable or disable line wrap within textbox.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bWrapEn</i>	Enable line wrap if true

#### Returns

none

### 9.38.2 Variable Documentation

#### 9.38.2.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

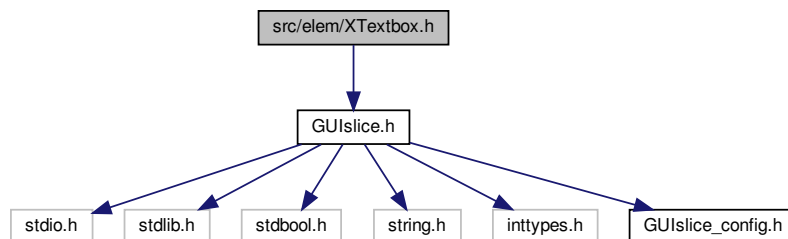
#### 9.38.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

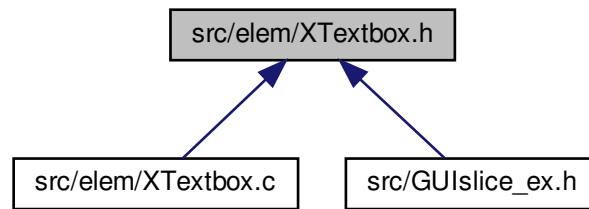
## 9.39 src/elem/XTextbox.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XTextbox.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXTextbox](#)  
*Extended data for Textbox element.*

## Macros

- `#define` [GSLC\\_TYPEX\\_TEXTBOX](#)
- `#define` [GSLC\\_XTEXTBOX\\_CODE\\_COL\\_SET](#)  
*Definitions for textbox special inline codes.*
- `#define` [GSLC\\_XTEXTBOX\\_CODE\\_COL\\_RESET](#)
- `#define` [XTEXTBOX\\_REDRAW\\_NONE](#)
- `#define` [XTEXTBOX\\_REDRAW\\_ALL](#)

## Functions

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemXTextboxCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsXTextbox](#) \*pXData, [gslc\\_tsRect](#) rElem, int16\_t nFontId, char \*pBuf, uint16\_t nBufRows, uint16\_t nBufCols)  
*Create a Textbox Element.*
- void [gslc\\_ElemXTextboxReset](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Reset the contents of the textbox.*
- bool [gslc\\_ElemXTextboxDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a Textbox element on the screen.*
- void [gslc\\_ElemXTextboxAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, char \*pTxt)  
*Add a text string to the textbox.*
- void [gslc\\_ElemXTextboxColSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) nCol)  
*Insert a color set code into the current buffer position.*
- void [gslc\\_ElemXTextboxColReset](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Insert a color reset code into the current buffer position.*
- void [gslc\\_ElemXTextboxWrapSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bWrapEn)  
*Enable or disable line wrap within textbox.*
- void [gslc\\_ElemXTextboxScrollSet](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nScrollPos, uint8\_t nScrollMax)  
*Set the textbox scroll position (nScrollPos) as a fraction of nScrollMax.*

## 9.39.1 Macro Definition Documentation

### 9.39.1.1 GSLC\_TYPEX\_TEXTBOX

```
#define GSLC_TYPEX_TEXTBOX
```

### 9.39.1.2 GSLC\_XTEXTBOX\_CODE\_COL\_RESET

```
#define GSLC_XTEXTBOX_CODE_COL_RESET
```

### 9.39.1.3 GSLC\_XTEXTBOX\_CODE\_COL\_SET

```
#define GSLC_XTEXTBOX_CODE_COL_SET
```

Definitions for textbox special inline codes.

### 9.39.1.4 XTEXTBOX\_REDRAW\_ALL

```
#define XTEXTBOX_REDRAW_ALL
```

### 9.39.1.5 XTEXTBOX\_REDRAW\_NONE

```
#define XTEXTBOX_REDRAW_NONE
```

## 9.39.2 Function Documentation

### 9.39.2.1 gslc\_ElemXTextboxAdd()

```
void gslc_ElemXTextboxAdd (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    char * pTxt )
```

Add a text string to the textbox.

- If it includes a newline then the buffer will advance to the next row
- If wrap has been enabled, then a newline will be forced

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>pTxt</i>	Pointer to text string (null-terminated)

**Returns**

none

**9.39.2.2 gslc\_ElemXTextboxColReset()**

```
void gslc_ElemXTextboxColReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Insert a color reset code into the current buffer position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

none

**9.39.2.3 gslc\_ElemXTextboxColSet()**

```
void gslc_ElemXTextboxColSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsColor nCol )
```

Insert a color set code into the current buffer position.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nCol</i>	Color to assign for next text written to textbox

**Returns**

none

**9.39.2.4 gslc\_ElemXTextboxCreate()**

```

gslc_tsElemRef* gslc_ElemXTextboxCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTextbox * pXData,
    gslc_tsRect rElem,
    int16_t nFontId,
    char * pBuf,
    uint16_t nBufRows,
    uint16_t nBufCols )

```

Create a Textbox Element.

- The textbox is a scrolling window designed for displaying multi-line text using a monospaced font. A character buffer is defined by nBufRows\*nBufCols to capture the added text. If the allocation buffer is larger than the display size (defined by rElem), then a scrollbar will be shown.
- Support for changing color within a row can be enabled with GSLC\_FEATURE\_XTEXTBOX\_EMBED 1
- Note that each color change command will consume 4 of the available "column" bytes.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining textbox size
in	<i>nFontId</i>	Font ID to use for text area
in	<i>pBuf</i>	Ptr to text buffer (already allocated) with size (nBufRows*nBufCols) chars
in	<i>nBufRows</i>	Number of rows in buffer
in	<i>nBufCols</i>	Number of columns in buffer (incl special codes)

**Returns**

Pointer to Element reference or NULL if failure

**9.39.2.5 gslc\_ElemXTextboxDraw()**

```

bool gslc_ElemXTextboxDraw (
    void * pvGui,

```

```
void * pvElemRef,
gslc_teRedrawType eRedraw )
```

Draw a Textbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

#### 9.39.2.6 [gslc\\_ElemXTextboxReset\(\)](#)

```
void gslc_ElemXTextboxReset (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Reset the contents of the textbox.

- Clears the buffer and resets the position

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

#### Returns

none

#### 9.39.2.7 [gslc\\_ElemXTextboxScrollSet\(\)](#)

```
void gslc_ElemXTextboxScrollSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    uint8_t nScrollPos,
    uint8_t nScrollMax )
```

Set the textbox scroll position (`nScrollPos`) as a fraction of `nScrollMax`.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nScrollPos</i>	New scroll position
in	<i>nScrollMax</i>	Maximum scroll position

## Returns

none

## 9.39.2.8 gslc\_ElemXTextboxWrapSet()

```
void gslc_ElemXTextboxWrapSet (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bWrapEn )
```

Enable or disable line wrap within textbox.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bWrapEn</i>	Enable line wrap if true

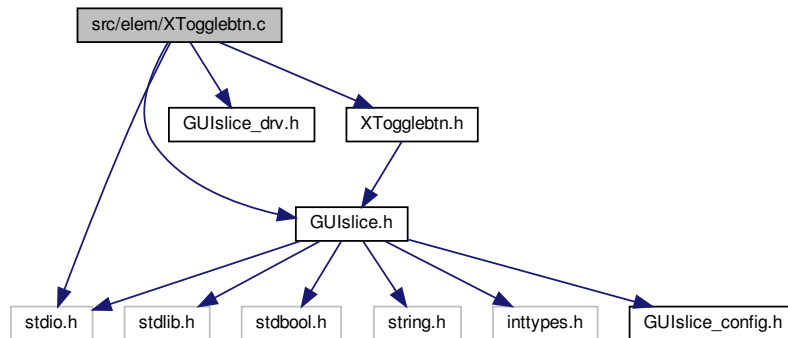
## Returns

none

## 9.40 src/elem/XTogglebtn.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "XTogglebtn.h"
#include <stdio.h>
```

Include dependency graph for XTogglebtn.c:



## Functions

- `gslc_tsElemRef * gslc_ElemXTogglebtnCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXTogglebtn *pXData, gslc\_tsRect rElem, gslc\_tsColor colThumb, gslc\_tsColor colOnState, gslc\_tsColor colOffState, bool bCircular, bool bChecked, GSLC\_CB\_TOUCH cbTouch)`  
Create a Togglebtn button Element.
- `bool gslc\_ElemXTogglebtnGetState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
Get a Togglebtn element's current state.
- `void gslc\_ElemXTogglebtnSetStateHelp (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bOn)`
- `void gslc\_ElemXTogglebtnSetState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bOn)`  
Set a Togglebtn element's current state.
- `void gslc\_ElemXTogglebtnToggleState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
Toggle a Togglebtn element's current state.
- `void gslc\_ElemXTogglebtnDrawCircularHelp (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsXTogglebtn *pTogglebtn)`
- `void gslc\_ElemXTogglebtnDrawRectangularHelp (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, gslc\_tsXTogglebtn *pTogglebtn)`
- `bool gslc\_ElemXTogglebtnDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
Draw a Togglebtn element on the screen.
- `bool gslc\_ElemXTogglebtnTouch (void *pvGui, void *pvElemRef, gslc\_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
Handle touch events to Togglebtn element.
- `gslc\_tsElemRef * gslc\_ElemXTogglebtnFindSelected (gslc_tsGui *pGui, int16_t nGroupId)`  
Find the togglebtn within a group that has been selected.

## Variables

- `const char GSLC\_PMEM\_ERRSTR\_NULL []`
- `const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL []`

### 9.40.1 Function Documentation



9.40.1.1 `gslc_ElemXTogglebtnCreate()`

```

gslc_tsElemRef* gslc_ElemXTogglebtnCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTogglebtn * pXData,
    gslc_tsRect rElem,
    gslc_tsColor colThumb,
    gslc_tsColor colOnState,
    gslc_tsColor colOffState,
    bool bCircular,
    bool bChecked,
    GSLC_CB_TOUCH cbTouch )

```

Create a Togglebtn button Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining togglebtn size
in	<i>colThumb</i>	Color of thumb
in	<i>colOnState</i>	Color to indicate on position
in	<i>colOffState</i>	Color to indicate off position
in	<i>bCircular</i>	Style of the toggle button circular or rectangular
in	<i>bChecked</i>	Default state
in	<i>cbTouch</i>	Callback for touch events

## Returns

Pointer to Element reference or NULL if failure

9.40.1.2 `gslc_ElemXTogglebtnDraw()`

```

bool gslc_ElemXTogglebtnDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )

```

Draw a Togglebtn element on the screen.

- Called from `gslc_ElemDraw()`

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.40.1.3 gslc\_ElemXTogglebtnDrawCircularHelp()**

```
void gslc_ElemXTogglebtnDrawCircularHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsXTogglebtn * pTogglebtn )
```

**9.40.1.4 gslc\_ElemXTogglebtnDrawRectangularHelp()**

```
void gslc_ElemXTogglebtnDrawRectangularHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    gslc_tsXTogglebtn * pTogglebtn )
```

**9.40.1.5 gslc\_ElemXTogglebtnFindSelected()**

```
gslc_tsElemRef* gslc_ElemXTogglebtnFindSelected (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```

Find the togglebtn within a group that has been selected.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n<sub>↔</sub></i> <i>GroupId</i>	Group ID to search

**Returns**

Element Ptr or NULL if none selected

#### 9.40.1.6 gslc\_ElemXTogglebtnGetState()

```
bool gslc_ElemXTogglebtnGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Togglebtn element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

##### Returns

Current state

#### 9.40.1.7 gslc\_ElemXTogglebtnSetState()

```
void gslc_ElemXTogglebtnSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

Set a Togglebtn element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bOn</i>	New state

##### Returns

none

#### 9.40.1.8 gslc\_ElemXTogglebtnSetStateHelp()

```
void gslc_ElemXTogglebtnSetStateHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

#### 9.40.1.9 `gslc_ElemXTogglebtnToggleState()`

```
void gslc_ElemXTogglebtnToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a Togglebtn element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

##### Returns

none

#### 9.40.1.10 `gslc_ElemXTogglebtnTouch()`

```
bool gslc_ElemXTogglebtnTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Togglebtn element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

##### Parameters

in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

##### Returns

true if success, false otherwise

## 9.40.2 Variable Documentation

## 9.40.2.1 ERRSTR\_NULL

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

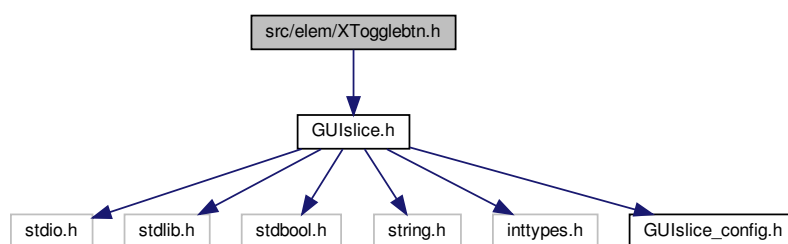
## 9.40.2.2 ERRSTR\_PXD\_NULL

```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```

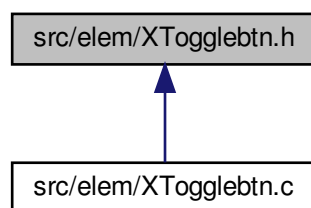
## 9.41 src/elem/XTogglebtn.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XTogglebtn.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gslc\\_tsXTogglebtn](#)

*Extended data for Togglebtn element.*

## Macros

- `#define GSLC_TYPEX_TOGGLEBTN`
- `#define gslc_ElemXTogglebtnCreate_P(pGui, nElemId, nPage, nX, nY, nW, nH, colThumb_, colOnState_, colOffState_, bCircular_, bChecked_, cbTouch)`  
*Create a Togglebtn button Element.*

## Functions

- `gslc_tsElemRef * gslc_ElemXTogglebtnCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsXTogglebtn *pXData, gslc_tsRect rElem, gslc_tsColor colThumb, gslc_tsColor colOnState, gslc_tsColor colOffState, bool bCircular, bool bChecked, GSLC_CB_TOUCH cbTouch)`  
*Create a Togglebtn button Element.*
- `bool gslc_ElemXTogglebtnGetState (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`  
*Get a Togglebtn element's current state.*
- `void gslc_ElemXTogglebtnSetState (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bOn)`  
*Set a Togglebtn element's current state.*
- `void gslc_ElemXTogglebtnToggleState (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`  
*Toggle a Togglebtn element's current state.*
- `bool gslc_ElemXTogglebtnDraw (void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`  
*Draw a Togglebtn element on the screen.*
- `bool gslc_ElemXTogglebtnTouch (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
*Handle touch events to Togglebtn element.*
- `gslc_tsElemRef * gslc_ElemXTogglebtnFindSelected (gslc_tsGui *pGui, int16_t nGroupId)`  
*Find the togglebtn within a group that has been selected.*

### 9.41.1 Macro Definition Documentation

#### 9.41.1.1 gslc\_ElemXTogglebtnCreate\_P

```
#define gslc_ElemXTogglebtnCreate_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    colThumb_,  
    colOnState_,  
    colOffState_,  
    bCircular_,  
    bChecked_,  
    cbTouch )
```

Create a Togglebtn button Element.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>colThumb</i> _	Color of thumb
in	<i>colOn</i> <i>State</i> _	Color to indicate on position
in	<i>colOff</i> <i>State</i> _	Color to indicate off position
in	<i>bCircular</i> _	Style of the toggle button circular or rectangular
in	<i>bChecked</i> _	Default state
in	<i>cbTouch</i>	Callback for touch events

## Returns

none

## 9.41.1.2 GSLC\_TYPEX\_TOGGLEBTN

```
#define GSLC_TYPEX_TOGGLEBTN
```

## 9.41.2 Function Documentation

## 9.41.2.1 gslc\_ElemXTogglebtnCreate()

```
gslc_tsElemRef* gslc_ElemXTogglebtnCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXTogglebtn * pXData,
    gslc_tsRect rElem,
    gslc_tsColor colThumb,
    gslc_tsColor colOnState,
    gslc_tsColor colOffState,
    bool bCircular,
    bool bChecked,
    GSLC_CB_TOUCH cbTouch )
```

Create a Togglebtn button Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining togglebtn size
in	<i>colThumb</i>	Color of thumb
in	<i>colOnState</i>	Color to indicate on position
in	<i>colOffState</i>	Color to indicate off position
in	<i>bCircular</i>	Style of the toggle button circular or rectangular
in	<i>bChecked</i>	Default state
in	<i>cbTouch</i>	Callback for touch events

**Returns**

Pointer to Element reference or NULL if failure

**9.41.2.2 gslc\_ElemXTogglebtnDraw()**

```
bool gslc_ElemXTogglebtnDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a Togglebtn element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.41.2.3 gslc\_ElemXTogglebtnFindSelected()**

```
gslc_tsElemRef* gslc_ElemXTogglebtnFindSelected (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```



Find the togglebtn within a group that has been selected.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ GroupId</i>	Group ID to search

**Returns**

Element Ptr or NULL if none selected

**9.41.2.4 gslc\_ElemXTogglebtnGetState()**

```
bool gslc_ElemXTogglebtnGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a Togglebtn element's current state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Current state

**9.41.2.5 gslc\_ElemXTogglebtnSetState()**

```
void gslc_ElemXTogglebtnSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

Set a Togglebtn element's current state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bOn</i>	New state

**Returns**

none

**9.41.2.6 gslc\_ElemXTogglebtnToggleState()**

```
void gslc_ElemXTogglebtnToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a Togglebtn element's current state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

none

**9.41.2.7 gslc\_ElemXTogglebtnTouch()**

```
bool gslc_ElemXTogglebtnTouch (
    void * pvGui,
    void * pvElemRef,
    gslc_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to Togglebtn element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

**Parameters**

in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

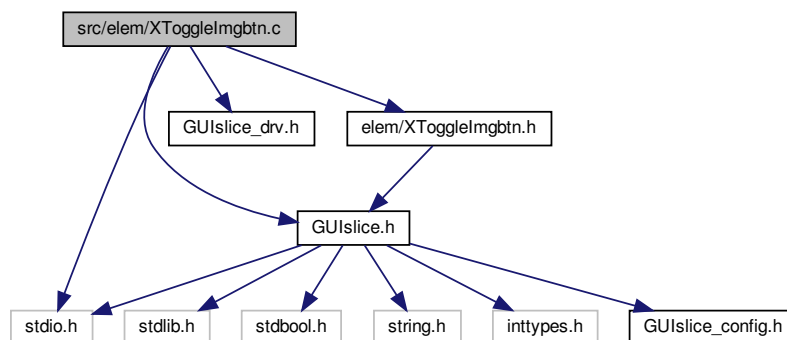
## Returns

true if success, false otherwise

## 9.42 src/elem/XToggleImgbtn.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XToggleImgbtn.h"
#include <stdio.h>
```

Include dependency graph for XToggleImgbtn.c:



## Functions

- [gslc\\_tsElemRef \\* gslc\\_ElemXToggleImgbtnCreate](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nElemId, [int16\\_t](#) nPage, [gslc\\_tsXToggleImgbtn](#) \*pXData, [gslc\\_tsRect](#) rElem, [gslc\\_tsImgRef](#) slmgRef, [gslc\\_tsImgRef](#) slmgRefSel, [bool](#) bOn, [GSLC\\_CB\\_TOUCH](#) cbTouch)  
*Create a ToggleImgbtn button Element.*
- [bool](#) [gslc\\_ElemXToggleImgbtnGetState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get a ToggleImgbtn element's current state.*
- [void](#) [gslc\\_ElemXToggleImgbtnSetStateHelp](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [bool](#) bOn)
- [void](#) [gslc\\_ElemXToggleImgbtnSetState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [bool](#) bOn)  
*Set a ToggleImgbtn element's current state.*
- [void](#) [gslc\\_ElemXToggleImgbtnToggleState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Toggle a ToggleImgbtn element's current state.*
- [bool](#) [gslc\\_ElemXToggleImgbtnDraw](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Draw a ToggleImgbtn element on the screen.*
- [bool](#) [gslc\\_ElemXToggleImgbtnTouch](#) ([void](#) \*pvGui, [void](#) \*pvElemRef, [gslc\\_teTouch](#) eTouch, [int16\\_t](#) nRelX, [int16\\_t](#) nRelY)  
*Handle touch events to ToggleImgbtn element.*
- [gslc\\_tsElemRef \\*](#) [gslc\\_ElemXToggleImgbtnFindSelected](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nGroupId)  
*Find the togglebtn within a group that has been selected.*

## Variables

- [const char](#) [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- [const char](#) [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

## 9.42.1 Function Documentation

### 9.42.1.1 gslc\_ElemXToggleImgbtnCreate()

```
gslc_tsElemRef* gslc_ElemXToggleImgbtnCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXToggleImgbtn * pXData,
    gslc_tsRect rElem,
    gslc_tsImgRef sImgRef,
    gslc_tsImgRef sImgRefSel,
    bool bOn,
    GSLC_CB_TOUCH cbTouch )
```

Create a ToggleImgbtn button Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining togglebtn size
in	<i>sImgRef</i>	Image reference to load (unselected state)
in	<i>sImgRefSel</i>	Image reference to load (selected state)
in	<i>bOn</i>	Default state, On or Off
in	<i>cbTouch</i>	Callback for touch events

#### Returns

Pointer to Element reference or NULL if failure

### 9.42.1.2 gslc\_ElemXToggleImgbtnDraw()

```
bool gslc_ElemXToggleImgbtnDraw (
    void * pvGui,
    void * pvElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a ToggleImgbtn element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pVGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.42.1.3 gslc\_ElemXToggleImgbtnFindSelected()**

```
gslc_tsElemRef* gslc_ElemXToggleImgbtnFindSelected (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```

Find the togglebtn within a group that has been selected.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ GroupId</i>	Group ID to search

**Returns**

Element Ptr or NULL if none selected

**9.42.1.4 gslc\_ElemXToggleImgbtnGetState()**

```
bool gslc_ElemXToggleImgbtnGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a ToggleImgbtn element's current state.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

**Returns**

Current state

#### 9.42.1.5 gslc\_ElemXToggleImgbtnSetState()

```
void gslc_ElemXToggleImgbtnSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

Set a ToggleImgbtn element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bOn</i>	New state

##### Returns

none

#### 9.42.1.6 gslc\_ElemXToggleImgbtnSetStateHelp()

```
void gslc_ElemXToggleImgbtnSetStateHelp (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

#### 9.42.1.7 gslc\_ElemXToggleImgbtnToggleState()

```
void gslc_ElemXToggleImgbtnToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a ToggleImgbtn element's current state.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

##### Returns

none

### 9.42.1.8 gslc\_ElemXToggleImgbtnTouch()

```
bool gslc_ElemXToggleImgbtnTouch (
    void * pVGui,
    void * pVElemRef,
    gslc\_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to ToggleImgbtn element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

## 9.42.2 Variable Documentation

### 9.42.2.1 ERRSTR\_NULL

```
const char GSLC\_PMEM ERRSTR_NULL[ ]
```

### 9.42.2.2 ERRSTR\_PXD\_NULL

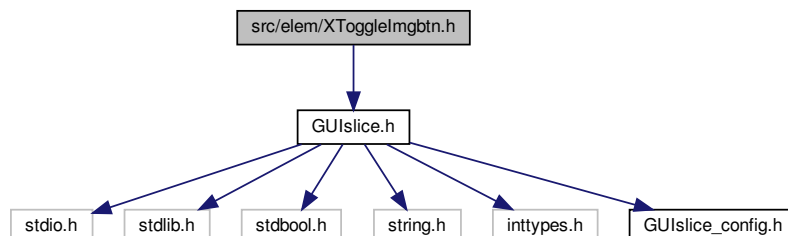
```
const char GSLC\_PMEM ERRSTR_PXD_NULL[ ]
```



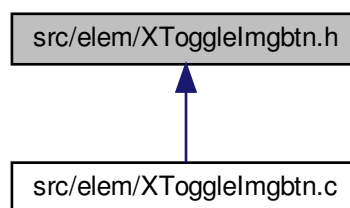
## 9.43 src/elem/XToggleImgbtn.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XToggleImgbtn.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gslc\\_tsXToggleImgbtn](#)

*Extended data for ToggleImgbtn element.*

### Macros

- #define [GSLC\\_TYPEX\\_TOGGLEIMGBTN](#)
- #define [gslc\\_ElemXToggleImgbtnCreate\\_P](#)(pGui, nElemId, nPage, nX, nY, nW, nH, slmgRef\_, slmgRefSel↵\_, bOn\_, cbTouch)

*Create a ToggleImgbtn button Element.*

## Functions

- `gslc_tsElemRef * gslc_ElemXToggleImgbtnCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc\_tsXToggleImgbtn *pXData, gslc\_tsRect rElem, gslc\_tsImgRef sImgRef, gslc\_tsImgRef sImgRefSel, bool bOn, GSLC\_CB\_TOUCH cbTouch)`  
Create a ToggleImgbtn button Element.
- `bool gslc_ElemXToggleImgbtnGetState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
Get a ToggleImgbtn element's current state.
- `void gslc_ElemXToggleImgbtnSetState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef, bool bOn)`  
Set a ToggleImgbtn element's current state.
- `void gslc_ElemXToggleImgbtnToggleState (gslc_tsGui *pGui, gslc\_tsElemRef *pElemRef)`  
Toggle a ToggleImgbtn element's current state.
- `bool gslc_ElemXToggleImgbtnDraw (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)`  
Draw a ToggleImgbtn element on the screen.
- `bool gslc_ElemXToggleImgbtnTouch (void *pvGui, void *pvElemRef, gslc\_teTouch eTouch, int16_t nRelX, int16_t nRelY)`  
Handle touch events to ToggleImgbtn element.
- `gslc\_tsElemRef * gslc_ElemXToggleImgbtnFindSelected (gslc_tsGui *pGui, int16_t nGroupId)`  
Find the togglebtn within a group that has been selected.

### 9.43.1 Macro Definition Documentation

#### 9.43.1.1 `gslc_ElemXToggleImgbtnCreate_P`

```
#define gslc_ElemXToggleImgbtnCreate_P(  
    pGui,  
    nElemId,  
    nPage,  
    nX,  
    nY,  
    nW,  
    nH,  
    sImgRef_,  
    sImgRefSel_,  
    bOn_,  
    cbTouch )
```

Create a ToggleImgbtn button Element.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or <code>GSLC_ID_AUTO</code> to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>nX</i>	X coordinate of element
in	<i>nY</i>	Y coordinate of element
in	<i>nW</i>	Width of element
in	<i>nH</i>	Height of element
in	<i>sImgRef_</i>	Image reference to load (unselected state)
in	<i>sImgRefSel_</i>	Image reference to load (selected state)
	<i>Sel_</i>	
in	<i>bOn_</i>	Default state, On or Off
in	<i>cbTouch</i>	Callback for touch events

**Returns**

none

**9.43.1.2 GSLC\_TYPEX\_TOGGLEIMGBTN**

```
#define GSLC_TYPEX_TOGGLEIMGBTN
```

**9.43.2 Function Documentation****9.43.2.1 gslc\_ElemXToggleImgbtnCreate()**

```
gslc_tsElemRef* gslc_ElemXToggleImgbtnCreate (
    gslc_tsGui * pGui,
    int16_t nElemId,
    int16_t nPage,
    gslc_tsXToggleImgbtn * pXData,
    gslc_tsRect rElem,
    gslc_tsImgRef sImgRef,
    gslc_tsImgRef sImgRefSel,
    bool bOn,
    GSLC_CB_TOUCH cbTouch )
```

Create a ToggleImgbtn button Element.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nElemId</i>	Element ID to assign (0..16383 or GSLC_ID_AUTO to autogen)
in	<i>nPage</i>	Page ID to attach element to
in	<i>pXData</i>	Ptr to extended element data structure
in	<i>rElem</i>	Rectangle coordinates defining togglebtn size
in	<i>sImgRef</i>	Image reference to load (unselected state)
in	<i>sImgRefSel</i>	Image reference to load (selected state)
in	<i>bOn</i>	Default state, On or Off
in	<i>cbTouch</i>	Callback for touch events

**Returns**

Pointer to Element reference or NULL if failure

### 9.43.2.2 `gslc_ElemXToggleImgbtnDraw()`

```
bool gslc_ElemXToggleImgbtnDraw (
    void * pVGui,
    void * pVElemRef,
    gslc_teRedrawType eRedraw )
```

Draw a ToggleImgbtn element on the screen.

- Called from `gslc_ElemDraw()`

#### Parameters

in	<i>pVGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pVElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

#### Returns

true if success, false otherwise

### 9.43.2.3 `gslc_ElemXToggleImgbtnFindSelected()`

```
gslc_tsElemRef* gslc_ElemXToggleImgbtnFindSelected (
    gslc_tsGui * pGui,
    int16_t nGroupId )
```

Find the togglebtn within a group that has been selected.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>n</i> ↔ <i>GroupId</i>	Group ID to search

#### Returns

Element Ptr or NULL if none selected

### 9.43.2.4 `gslc_ElemXToggleImgbtnGetState()`

```
bool gslc_ElemXToggleImgbtnGetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Get a ToggleImgbtn element's current state.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

Current state

## 9.43.2.5 gslc\_ElemXToggleImgbtnSetState()

```
void gslc_ElemXToggleImgbtnSetState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef,
    bool bOn )
```

Set a ToggleImgbtn element's current state.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>bOn</i>	New state

## Returns

none

## 9.43.2.6 gslc\_ElemXToggleImgbtnToggleState()

```
void gslc_ElemXToggleImgbtnToggleState (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

Toggle a ToggleImgbtn element's current state.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference

## Returns

none

### 9.43.2.7 gslc\_ElemXToggleImgbtnTouch()

```
bool gslc_ElemXToggleImgbtnTouch (
    void * pvGui,
    void * pvElemRef,
    gslc\_teTouch eTouch,
    int16_t nRelX,
    int16_t nRelY )
```

Handle touch events to ToggleImgbtn element.

- Called from [gslc\\_ElemSendEventTouch\(\)](#)

#### Parameters

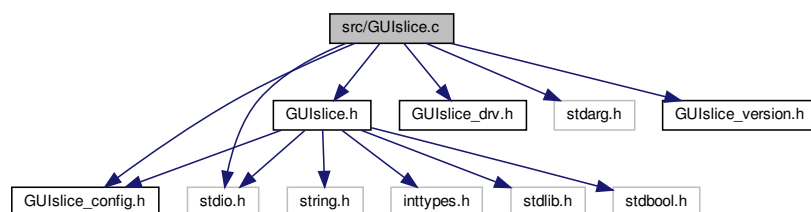
in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element reference (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

#### Returns

true if success, false otherwise

## 9.44 src/GUISlice.c File Reference

```
#include "GUISlice_config.h"
#include "GUISlice.h"
#include "GUISlice_drv.h"
#include <stdio.h>
#include <stdarg.h>
#include "GUISlice_version.h"
Include dependency graph for GUISlice.c:
```



## Enumerations

- enum [gslc\\_tsDebugPrintState](#) {  
[GSLC\\_S\\_DEBUG\\_PRINT\\_NORM](#), [GSLC\\_S\\_DEBUG\\_PRINT\\_TOKEN](#), [GSLC\\_S\\_DEBUG\\_PRINT\\_UINT16](#),  
[GSLC\\_S\\_DEBUG\\_PRINT\\_CHAR](#),  
[GSLC\\_S\\_DEBUG\\_PRINT\\_STR](#), [GSLC\\_S\\_DEBUG\\_PRINT\\_STR\\_P](#) }

## Functions

- char \* [gslc\\_GetVer](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice version number.*
- const char \* [gslc\\_GetNameDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice display driver name.*
- const char \* [gslc\\_GetNameTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice touch driver name.*
- void \* [gslc\\_GetDriverDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native display driver instance.*
- void \* [gslc\\_GetDriverTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native touch driver instance.*
- bool [gslc\\_Init](#) ([gslc\\_tsGui](#) \*pGui, void \*pvDriver, [gslc\\_tsPage](#) \*asPage, uint8\_t nMaxPage, [gslc\\_tsFont](#) \*asFont, uint8\_t nMaxFont)  
*Initialize the GUISlice library.*
- void [gslc\\_SetPinPollFunc](#) ([gslc\\_tsGui](#) \*pGui, [GSLC\\_CB\\_PIN\\_POLL](#) pfunc)  
*Specify the callback function that is used to collect the state of any external inputs (eg.*
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)  
*Specify the mapping between external pin inputs (fetched by the SetPinPollFunc() callback and the GUI actions.*
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_tsAction](#) eAction, int16\_t nActionVal)  
*Add an entry into the external input mapping table.*
- bool [gslc\\_InputMapLookup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_tsAction](#) \*peAction, int16\_t \*pnActionVal)  
*Lookup an entry in the external input mapping table.*
- void [gslc\\_InitDebug](#) ([GSLC\\_CB\\_DEBUG\\_OUT](#) pfunc)  
*Initialize debug output.*
- void [gslc\\_DebugPrintf](#) (const char \*pFmt,...)  
*Optimized printf routine for GUISlice debug/error output.*
- void [gslc\\_Quit](#) ([gslc\\_tsGui](#) \*pGui)  
*Exit the GUISlice environment.*
- void [gslc\\_Update](#) ([gslc\\_tsGui](#) \*pGui)  
*Perform main GUISlice handling functions.*
- [gslc\\_tsEvent](#) [gslc\\_EventCreate](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsEventType](#) eType, uint8\_t nSubType, void \*pvScope, void \*pvData)  
*Create an event structure.*
- bool [gslc\\_IsInRect](#) (int16\_t nSelX, int16\_t nSelY, [gslc\\_tsRect](#) rRect)  
*Determine if a coordinate is inside of a rectangular region.*
- bool [gslc\\_IsInWH](#) (int16\_t nSelX, int16\_t nSelY, uint16\_t nWidth, uint16\_t nHeight)  
*Determine if a coordinate is inside of a width x height region.*
- void [gslc\\_OrderCoord](#) (int16\_t \*pnX0, int16\_t \*pnY0, int16\_t \*pnX1, int16\_t \*pnY1)  
*Order coordinates for clipping.*
- bool [gslc\\_ClipPt](#) ([gslc\\_tsRect](#) \*pClipRect, int16\_t nX, int16\_t nY)  
*Perform basic clipping of a single point to a clipping region.*
- bool [gslc\\_ClipLine](#) ([gslc\\_tsRect](#) \*pClipRect, int16\_t \*pnX0, int16\_t \*pnY0, int16\_t \*pnX1, int16\_t \*pnY1)  
*Perform basic clipping of a line to a clipping region.*

- `bool gslc_ClipRect (gslc_tsRect *pClipRect, gslc_tsRect *pRect)`  
*Perform basic clipping of a rectangle to a clipping region.*
- `gslc_tslmgRef gslc_ResetImage ()`  
*Create a blank image reference structure.*
- `gslc_tslmgRef gslc_GetImageFromFile (const char *pFname, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap file in LINUX filesystem.*
- `gslc_tslmgRef gslc_GetImageFromSD (const char *pFname, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap file in SD card.*
- `gslc_tslmgRef gslc_GetImageFromRam (unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap in SRAM.*
- `gslc_tslmgRef gslc_GetImageFromProg (const unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)`  
*Create an image reference to a bitmap in program memory (PROGMEM)*
- `int16_t gslc_sinFX (int16_t n64Ang)`  
*Calculate fixed-point sine function from fractional degrees.*
- `int16_t gslc_cosFX (int16_t n64Ang)`  
*Calculate fixed-point cosine function from fractional degrees.*
- `void gslc_PolarToXY (uint16_t nRad, int16_t n64Ang, int16_t *nDX, int16_t *nDY)`  
*Convert polar coordinate to cartesian.*
- `gslc_tsColor gslc_ColorBlend2 (gslc_tsColor colStart, gslc_tsColor colEnd, uint16_t nMidAmt, uint16_t nBlendAmt)`  
*Create a color based on a blend between two colors.*
- `gslc_tsColor gslc_ColorBlend3 (gslc_tsColor colStart, gslc_tsColor colMid, gslc_tsColor colEnd, uint16_t nMidAmt, uint16_t nBlendAmt)`  
*Create a color based on a blend between three colors.*
- `bool gslc_ColorEqual (gslc_tsColor a, gslc_tsColor b)`  
*Check whether two colors are equal.*
- `void gslc_DrawSetPixel (gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)`  
*Set a pixel on the active screen to the given color with lock.*
- `void gslc_DrawLine (gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)`  
*Draw an arbitrary line using Bresenham's algorithm.*
- `void gslc_DrawLineH (gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nW, gslc_tsColor nCol)`  
*Draw a horizontal line.*
- `void gslc_DrawLineV (gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nH, gslc_tsColor nCol)`  
*Draw a vertical line.*
- `void gslc_DrawLinePolar (gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nRadStart, uint16_t nRadEnd, int16_t n64Ang, gslc_tsColor nCol)`  
*Draw a polar ray segment.*
- `void gslc_DrawFrameRect (gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)`  
*Draw a framed rectangle.*
- `void gslc_DrawFrameRoundRect (gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)`  
*Draw a framed rounded rectangle.*
- `void gslc_DrawFillRect (gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)`  
*Draw a filled rectangle.*
- `void gslc_DrawFillRoundRect (gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)`  
*Draw a filled rounded rectangle.*
- `gslc_tsRect gslc_ExpandRect (gslc_tsRect rRect, int16_t nExpandW, int16_t nExpandH)`  
*Expand or contract a rectangle in width and/or height (equal amounts on both side), based on the centerpoint of the rectangle.*
- `void gslc_UnionRect (gslc_tsRect *pRect, gslc_tsRect rAddRect)`  
*Expand a rect to include another rect.*
- `void gslc_InvalidateRgnReset (gslc_tsGui *pGui)`



- Reset the invalidation region.*
- void [gslc\\_InvalidateRgnScreen](#) ([gslc\\_tsGui](#) \*pGui)
- Mark the entire screen as invalidated.*
- void [gslc\\_InvalidateRgnPage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage)
- Include an entire page (eg.*
- void [gslc\\_InvalidateRgnAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rAddRect)
- Add a rectangular region to the invalidation region.*
- void [gslc\\_DrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a framed circle.*
- void [gslc\\_DrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a filled circle.*
- void [gslc\\_DrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
- Draw a framed triangle.*
- void [gslc\\_SwapCoords](#) (int16\_t \*pnXa, int16\_t \*pnYa, int16\_t \*pnXb, int16\_t \*pnYb)
- void [gslc\\_DrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
- Draw a filled triangle.*
- void [gslc\\_DrawFrameQuad](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*psPt, [gslc\\_tsColor](#) nCol)
- Draw a framed quadrilateral.*
- void [gslc\\_DrawFillQuad](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*psPt, [gslc\\_tsColor](#) nCol)
- Draw a filled quadrilateral.*
- void [gslc\\_DrawFillSectorBase](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArcStart, [gslc\\_tsColor](#) cArcEnd, bool bGradient, int16\_t nAngGradStart, int16\_t nAngGradRange, int16\_t nAngSecStart, int16\_t nAngSecEnd)
- void [gslc\\_DrawFillGradSector](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArcStart, [gslc\\_tsColor](#) cArcEnd, int16\_t nAngSecStart, int16\_t nAngSecEnd, int16\_t nAngGradStart, int16\_t nAngGradRange)
- Draw a gradient filled sector of a circle with support for inner and outer radius.*
- void [gslc\\_DrawFillSector](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArc, int16\_t nAngSecStart, int16\_t nAngSecEnd)
- Draw a flat filled sector of a circle with support for inner and outer radius.*
- bool [gslc\\_FontSetBase](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nFontInd, int16\_t nFontId, [gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)
- bool [gslc\\_FontSet](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)
- Load a font into the local font cache and store as font ID (nFontId)*
- bool [gslc\\_FontAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)
- Load a font into the local font cache and assign font ID (nFontId).*
- [gslc\\_tsFont](#) \* [gslc\\_FontGet](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId)
- Fetch a font from its ID value.*
- bool [gslc\\_FontSetMode](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefMode](#) eFontMode)
- Set the font operating mode.*
- bool [gslc\\_PageEvent](#) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)
- Common event handler function for a page.*
- void [gslc\\_PageAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, [gslc\\_tsElem](#) \*psElem, uint16\_t nMaxElem, [gslc\\_tsElemRef](#) \*psElemRef, uint16\_t nMaxElemRef)
- Add a page to the GUI.*
- int [gslc\\_GetPageCur](#) ([gslc\\_tsGui](#) \*pGui)
- Fetch the current page ID.*

- void [gslc\\_SetStackPage](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nStackPos, int16\_t nPageld)
 

*Assign a page to the page stack.*
- void [gslc\\_SetStackState](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nStackPos, bool bActive, bool bDoDraw)
 

*Change the status of a page in a page stack.*
- void [gslc\\_SetPageBase](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld)
 

*Assigns a page for the base layer in the page stack.*
- void [gslc\\_SetPageCur](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld)
 

*Select a page for the current layer in the page stack.*
- void [gslc\\_SetPageOverlay](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld)
 

*Select a page for the overlay layer in the page stack.*
- void [gslc\\_PopupShow](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld, bool bModal)
 

*Show a popup dialog.*
- void [gslc\\_PopupHide](#) ([gslc\\_tsGui](#) \*pGui)
 

*Hides the currently active popup dialog.*
- void [gslc\\_PageRedrawSet](#) ([gslc\\_tsGui](#) \*pGui, bool bRedraw)
 

*Update the need-redraw status for the current page.*
- bool [gslc\\_PageRedrawGet](#) ([gslc\\_tsGui](#) \*pGui)
 

*Get the need-redraw status for the current page.*
- void [gslc\\_PageRedrawCalc](#) ([gslc\\_tsGui](#) \*pGui)
 

*Perform a redraw calculation on the page to determine if additional elements should also be redrawn.*
- void [gslc\\_PageRedrawGo](#) ([gslc\\_tsGui](#) \*pGui)
 

*Redraw all elements on the active page.*
- void [gslc\\_PageFlipSet](#) ([gslc\\_tsGui](#) \*pGui, bool bNeeded)
 

*Indicate whether the screen requires page flip.*
- bool [gslc\\_PageFlipGet](#) ([gslc\\_tsGui](#) \*pGui)
 

*Get state of pending page flip state.*
- void [gslc\\_PageFlipGo](#) ([gslc\\_tsGui](#) \*pGui)
 

*Update the visible screen if page has been marked for flipping.*
- [gslc\\_tsPage](#) \* [gslc\\_PageFindByld](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld)
 

*Find a page in the GUI by its ID.*
- [gslc\\_tsElemRef](#) \* [gslc\\_PageFindElemByld](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageld, int16\_t nElemld)
 

*Find an element in the GUI by its Page ID and Element ID.*
- int [gslc\\_ElemGetld](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
 

*Get an Element ID from an element structure.*
- uint8\_t [gslc\\_GetElemRefFlag](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nFlagMask)
 

*Get the flags associated with an element reference.*
- void [gslc\\_SetElemRefFlag](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, uint8\_t nFlagMask, uint8\_t nFlagVal)
 

*Set the flags associated with an element reference.*
- [gslc\\_tsElem](#) \* [gslc\\_GetElemFromRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
 

*Returns a pointer to an element from an element reference, copying from FLASH to RAM if element is stored in PROGMEM.*
- [gslc\\_tsElem](#) \* [gslc\\_GetElemFromRefD](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nLineNum)
 

*Returns a pointer to an element from an element reference.*
- void \* [gslc\\_GetXDataFromRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nType, int16\_t nLineNum)
 

*Returns a pointer to the data structure associated with an extended element.*
- void [gslc\\_SetRoundRadius](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRadius)
 

*Set the global rounded radius.*
- void [gslc\\_SetFocusCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) colFocusNone, [gslc\\_tsColor](#) colFocus, [gslc\\_tsColor](#) colFocusEdit)

*Set the global focus color choices.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)

*Create a Text Element.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId, [GSLC\\_CB\\_TOUCH](#) cbTouch)

*Create a textual Button Element.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, [gslc\\_tsImgRef](#) sImgRef, [gslc\\_tsImgRef](#) sImgRefSel, [GSLC\\_CB\\_TOUCH](#) cbTouch)

*Create a graphical Button Element.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBox](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem)

*Create a Box Element.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1)

*Create a Line Element.*

- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, [gslc\\_tsImgRef](#) sImgRef)

*Create an image Element.*

- bool [gslc\\_ElemEvent](#) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)

*Common event handler function for an element.*

- void [gslc\\_ElemDraw](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, int16\_t nElemId)

*Draw an element to the active display.*

- void [gslc\\_DrawTxtBase](#) ([gslc\\_tsGui](#) \*pGui, char \*pStrBuf, [gslc\\_tsRect](#) rTxt, [gslc\\_tsFont](#) \*pTxtFont, [gslc\\_teTxtFlags](#) eTxtFlags, int8\_t eTxtAlign, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg, int16\_t nMarginW, int16\_t nMarginH)

*Draw text with full text justification.*

- bool [gslc\\_ElemDrawByRef](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Draw an element to the active display.*

- void [gslc\\_ElemSetFillEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFillEn)

*Set the fill state for an Element.*

- void [gslc\\_ElemSetFrameEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFrameEn)

*Set the frame state for an Element.*

- void [gslc\\_ElemSetRoundEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bRoundEn)

*Set the rounded frame/fill state for an Element.*

- void [gslc\\_ElemSetCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colFrame, [gslc\\_tsColor](#) colFill, [gslc\\_tsColor](#) colFillGlow)

*Update the common color selection for an Element.*

- void [gslc\\_ElemSetGlowCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colFrameGlow, [gslc\\_tsColor](#) colFillGlow, [gslc\\_tsColor](#) colTxtGlow)

*Update the common color selection for glowing state of an Element.*

- void [gslc\\_ElemSetGroup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int nGroupId)

*Set the group ID for an element.*

- int [gslc\\_ElemGetGroup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the group ID for an element.*

- void [gslc\\_ElemSetRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRect](#) rElem)

*Set the position and size for an element.*

- [gslc\\_tsRect](#) [gslc\\_ElemGetRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the rectangular region for an element.*

- void [gslc\\_ElemSetTxtAlign](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, unsigned nAlign)

*Set the alignment of a textual element (horizontal and vertical)*

- void [gslc\\_ElemSetTxtMargin](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, unsigned nMargin)

*Set the margin around of a textual element.*

- void [gslc\\_ElemSetTxtMarginXY](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nMarginX, int8\_t nMarginY)

*Set the margin around of a textual element (X & Y offsets can be different)*

- void [gslc\\_StrCopy](#) (char \*pDstStr, const char \*pSrcStr, uint16\_t nDstLen)

*Helper routine to perform string deep copy.*

- void [gslc\\_ElemSetTxtStr](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStr)

*Update the text string associated with an Element.*

- char \* [gslc\\_ElemGetTxtStr](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Fetch the current text string associated with an Element.*

- void [gslc\\_ElemSetTxtCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colVal)

*Update the text string color associated with an Element ID.*

- void [gslc\\_ElemSetTxtMem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)

*Update the text string location in memory.*

- void [gslc\\_ElemSetTxtEnc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)

*Update the text string encoding mode.*

- void [gslc\\_ElemUpdateFont](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int nFontId)

*Update the Font selected for an Element's text.*

- void [gslc\\_ElemSetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)

*Update the need-redraw status for an element.*

- [gslc\\_teRedrawType](#) [gslc\\_ElemGetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the need-redraw status for an element.*

- void [gslc\\_ElemSetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowing)

*Update the glowing indicator for an element.*

- bool [gslc\\_ElemGetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the glowing indicator for an element.*

- void [gslc\\_ElemSetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocused)

*Update the focused indicator for an element.*

- bool [gslc\\_ElemGetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the focused indicator for an element.*

- bool [gslc\\_ElemGetEditEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

- void [gslc\\_ElemSetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bEditing)

*Update the editing indicator for an element.*

- bool [gslc\\_ElemGetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the editing indicator for an element.*

- void [gslc\\_ElemSetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bVisible)

*Update the visibility status for an element.*

- bool [gslc\\_ElemGetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the visibility status for an element.*

- bool [gslc\\_ElemGetOnScreen](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Determine whether an element is visible on the screen.*

- void [gslc\\_ElemSetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowEn)

*Update the glowing enable for an element.*

- bool [gslc\\_ElemGetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the glowing enable for an element.*

- bool [gslc\\_ElemGetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

*Get the focus enable for an element.*

- void [gslc\\_ElemSetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocusEn)

*Set the focus enable for an element.*

- void [gslc\\_ElemSetClickEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bClickEn)

*Update the click enable for an element.*

- void [gslc\\_ElemSetTouchFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_TOUCH](#) funcCb)  
*Update the touch function callback for an element.*
- void [gslc\\_ElemSetStyleFrom](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRefSrc, [gslc\\_tsElemRef](#) \*pElemRefDest)  
*Copy style settings from one element to another.*
- [int8\\_t](#) [gslc\\_ElemCalcResizeForFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Calculate the change in dimensions of an element to account for any change in focus and/or frame attributes.*
- void [gslc\\_ElemGrowRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [int8\\_t](#) nDelta)  
*Increase or decrease the size of an element's region.*
- void [gslc\\_ResetRectState](#) ([gslc\\_tsRectState](#) \*pState)  
*Reset the element region state struct.*
- void [gslc\\_ElemCalcRectState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRectState](#) \*pState)  
*Calculate the element region state struct.*
- void [gslc\\_ElemSetDrawFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_DRAW](#) funcCb)  
*Assign the drawing callback function for an element.*
- void [gslc\\_ElemSetTickFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_TICK](#) funcCb)  
*Assign the tick callback function for an element.*
- bool [gslc\\_ElemOwnsCoord](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [int16\\_t](#) nX, [int16\\_t](#) nY, bool bOnlyClickEn)  
*Determine if a coordinate is inside of an element.*
- void [gslc\\_CollectInput](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, [gslc\\_tsEventTouch](#) \*pEventTouch)  
*Handle direct input events within the element collection.*
- void [gslc\\_CollectTouch](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, [gslc\\_tsEventTouch](#) \*pEventTouch)  
*Handle touch events within the element collection.*
- bool [gslc\\_CollectTouchCompound](#) (void \*pvGui, void \*pvElemRef, [gslc\\_tsEventTouch](#) eTouch, [int16\\_t](#) nRelX, [int16\\_t](#) nRelY, [gslc\\_tsCollect](#) \*pCollect)  
*Handle dispatch of touch (up,down,move) events to compound elements sub elements.*
- bool [gslc\\_ElemCanFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, [int16\\_t](#) nElemInd)
- [gslc\\_tsElemRef](#) \* [gslc\\_FocusElemGet](#) ([gslc\\_tsGui](#) \*pGui)  
*Find the currently focused element.*
- void [gslc\\_FocusElemIndSet](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nPageInd, [int16\\_t](#) nElemInd, bool bFocus)  
*Change the focus to the indexed element on the specified page.*
- void [gslc\\_FocusSetToTrackedElem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect)  
*Change the focus to the currently-tracked element.*
- void [gslc\\_FocusPageStep](#) ([gslc\\_tsGui](#) \*pGui, bool bNext)  
*Advance the focus to the next page in the page stack.*
- [int16\\_t](#) [gslc\\_FocusElemStep](#) ([gslc\\_tsGui](#) \*pGui, bool bNext)  
*Advance the focus to the next element in the focused page.*
- void [gslc\\_TrackInput](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputRawEvent](#) eInputEvent, [int16\\_t](#) nInputVal)  
*Handles a direct input event and performs the necessary tracking, glowing and selection actions depending on the state.*
- void [gslc\\_TrackTouch](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage, [int16\\_t](#) nX, [int16\\_t](#) nY, [uint16\\_t](#) nPress)  
*Handles a touch event and performs the necessary tracking, glowing and selection actions depending on the press state.*
- bool [gslc\\_InitTouch](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)  
*Initialize the touchscreen device driver.*
- bool [gslc\\_GetTouch](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) \*pnX, [int16\\_t](#) \*pnY, [uint16\\_t](#) \*pnPress, [gslc\\_tsInputRawEvent](#) \*peInputEvent, [int16\\_t](#) \*pnInputVal)  
*Initialize the touchscreen device driver.*
- void [gslc\\_SetTouchRemapEn](#) ([gslc\\_tsGui](#) \*pGui, bool bEn)  
*Configure touchscreen remapping.*

- void [gslc\\_SetTouchRemapCal](#) ([gslc\\_tsGui](#) \*pGui, uint16\_t nXMin, uint16\_t nXMax, uint16\_t nYMin, uint16\_t nYMax)  
*Configure touchscreen calibration remapping values.*
- void [gslc\\_SetTouchPressCal](#) ([gslc\\_tsGui](#) \*pGui, uint16\_t nPressMin, uint16\_t nPressMax)  
*Configure touchscreen calibration pressure values.*
- void [gslc\\_SetTouchRemapYX](#) ([gslc\\_tsGui](#) \*pGui, bool bSwap)  
*Configure touchscreen XY swap.*
- void [gslc\\_SetTouchEn](#) ([gslc\\_tsGui](#) \*pGui, bool bEn)  
*Make touchscreen sensitive (GUI reacts to touch events) or insensitive (GUI ignores touch events)*
- bool [gslc\\_GetTouchEn](#) ([gslc\\_tsGui](#) \*pGui)  
*Get whether the GUI will react to touch events or not.*
- [gslc\\_tsElem](#) [gslc\\_ElemCreate](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPageId, int16\_t nType, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create a new element with default styling.*
- bool [gslc\\_CollectEvent](#) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)  
*Common event handler function for an element collection.*
- [gslc\\_tsElemRef](#) \* [gslc\\_CollectElemAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, const [gslc\\_tsElem](#) \*pElem, [gslc\\_teElemRefFlags](#) eFlags)  
*Add an element to a collection.*
- bool [gslc\\_CollectGetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect)  
*Determine if any elements in a collection need redraw.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, [gslc\\_tsElem](#) \*pElem, [gslc\\_teElemRefFlags](#) eFlags)  
*Add the Element to the list of generated elements in the GUI environment.*
- [gslc\\_tsRect](#) [gslc\\_GetClipRect](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the current the clipping rectangle.*
- bool [gslc\\_SetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for further drawing.*
- bool [gslc\\_ImgRefEqual](#) ([gslc\\_tsImgRef](#) \*pImgRef1, [gslc\\_tsImgRef](#) \*pImgRef2)
- void [gslc\\_ElemSetImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsImgRef](#) sImgRef, [gslc\\_tsImgRef](#) sImgRefSel)  
*Set an element to use a bitmap image.*
- bool [gslc\\_SetBkgndImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsImgRef](#) sImgRef)  
*Configure the background to use a bitmap image.*
- bool [gslc\\_SetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the background to use a solid color.*
- bool [gslc\\_SetTransparentColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the color to use for image transparency.*
- bool [gslc\\_GuiRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)  
*Dynamically change rotation, automatically adapt touchscreen axes swap/flip.*
- bool [gslc\\_ElemSendEventTouch](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRefTracked, [gslc\\_teTouch](#) eTouch, int16\_t nX, int16\_t nY)  
*Trigger an element's touch event.*
- void [gslc\\_ResetElem](#) ([gslc\\_tsElem](#) \*pElem)  
*Initialize an Element struct.*
- void [gslc\\_ResetFont](#) ([gslc\\_tsFont](#) \*pFont)  
*Initialize a Font struct.*
- void [gslc\\_ElemDestruct](#) ([gslc\\_tsElem](#) \*pElem)  
*Free up any members associated with an element.*
- void [gslc\\_CollectDestruct](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect)  
*Free up any members associated with an element collection.*

- void `gslc_PageDestruct` (`gslc_tsGui` \*pGui, `gslc_tsPage` \*pPage)  
*Free up any members associated with a page.*
- void `gslc_GuiDestruct` (`gslc_tsGui` \*pGui)  
*Free up any surfaces associated with the GUI, pages, collections and elements.*
- void `gslc_CollectReset` (`gslc_tsCollect` \*pCollect, `gslc_tsElem` \*asElem, `uint16_t` nElemMax, `gslc_tsElemRef` \*asElemRef, `uint16_t` nElemRefMax)  
*Reset the members of an element collection.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemById` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `int16_t` nElemId)  
*Find an element in a collection by its Element ID.*
- `int` `gslc_CollectGetNextId` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Allocate the next available Element ID in a collection.*
- `gslc_tsElemRef` \* `gslc_CollectGetElemRefTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Get the element within a collection that is currently being tracked.*
- void `gslc_CollectSetElemTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRef, `int16_t` nElemInd)  
*Set the element within a collection that is currently being tracked.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemFromCoord` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `int16_t` nX, `int16_t` nY, `int16_t` \*pnElemInd)  
*Find an element in a collection by a coordinate coordinate.*
- void `gslc_CollectSetParent` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRefParent)  
*Assign the parent element reference to all elements within a collection.*

## Variables

- `GSLC_CB_DEBUG_OUT` `g_pfDebugOut`  
*Global debug output function.*
- `const` `uint16_t` `m_nLUTSinFOX16` [257]
- `const` `char` `GSLC_PMEM_ERRSTR_NULL` []
- `const` `char` `GSLC_PMEM_ERRSTR_PXD_NULL` []

## 9.44.1 Enumeration Type Documentation

### 9.44.1.1 `gslc_teDebugPrintState`

`enum` `gslc_teDebugPrintState`

#### Enumerator

<code>GSLC_S_DEBUG_PRINT_NORM</code>	
<code>GSLC_S_DEBUG_PRINT_TOKEN</code>	
<code>GSLC_S_DEBUG_PRINT_UINT16</code>	
<code>GSLC_S_DEBUG_PRINT_CHAR</code>	
<code>GSLC_S_DEBUG_PRINT_STR</code>	
<code>GSLC_S_DEBUG_PRINT_STR_P</code>	



## 9.44.2 Function Documentation

### 9.44.2.1 gslc\_DrawFillSectorBase()

```
void gslc_DrawFillSectorBase (
    gslc_tsGui * pGui,
    int16_t nQuality,
    int16_t nMidX,
    int16_t nMidY,
    int16_t nRad1,
    int16_t nRad2,
    gslc_tsColor cArcStart,
    gslc_tsColor cArcEnd,
    bool bGradient,
    int16_t nAngGradStart,
    int16_t nAngGradRange,
    int16_t nAngSecStart,
    int16_t nAngSecEnd )
```

### 9.44.2.2 gslc\_ElemCanFocus()

```
bool gslc_ElemCanFocus (
    gslc_tsGui * pGui,
    gslc_tsCollect * pCollect,
    int16_t nElemInd )
```

### 9.44.2.3 gslc\_ElemGetEditEn()

```
bool gslc_ElemGetEditEn (
    gslc_tsGui * pGui,
    gslc_tsElemRef * pElemRef )
```

**Todo** Doc

### 9.44.2.4 gslc\_FontSetBase()

```
bool gslc_FontSetBase (
    gslc_tsGui * pGui,
    uint8_t nFontInd,
    int16_t nFontId,
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```



#### 9.44.2.5 `gslc_ImgRefEqual()`

```
bool gslc_ImgRefEqual (
    gslc_tsImgRef * pImgRef1,
    gslc_tsImgRef * pImgRef2 )
```

#### 9.44.2.6 `gslc_OrderCoord()`

```
void gslc_OrderCoord (
    int16_t * pnX0,
    int16_t * pnY0,
    int16_t * pnX1,
    int16_t * pnY1 )
```

#### 9.44.2.7 `gslc_SwapCoords()`

```
void gslc_SwapCoords (
    int16_t * pnXa,
    int16_t * pnYa,
    int16_t * pnXb,
    int16_t * pnYb )
```

### 9.44.3 Variable Documentation

#### 9.44.3.1 `ERRSTR_NULL`

```
const char ERRSTR_NULL[ ]
```

#### 9.44.3.2 `ERRSTR_PXD_NULL`

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

#### 9.44.3.3 `g_pfDebugOut`

```
GSLC_CB_DEBUG_OUT g_pfDebugOut
```

Global debug output function.

- The user assigns this function via `gslc_InitDebug()`

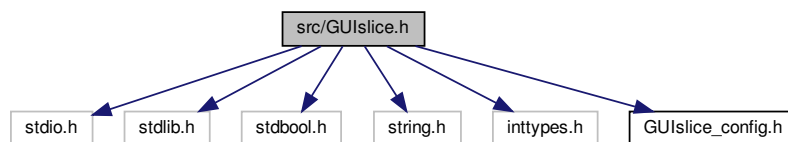
#### 9.44.3.4 m\_nLUTSinF0X16

```
const uint16_t m_nLUTSinF0X16
```

### 9.45 src/GUISlice.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <inttypes.h>
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [gslc\\_tsRect](#)  
*Rectangular region. Defines X,Y corner coordinates plus dimensions.*
- struct [gslc\\_tsPt](#)  
*Define point coordinates.*
- struct [gslc\\_tsColor](#)  
*Color structure. Defines RGB triplet.*
- struct [gslc\\_tsRectState](#)  
*State associated with an element's region.*
- struct [gslc\\_tsEvent](#)  
*Event structure.*
- struct [gslc\\_tsEventTouch](#)  
*Structure used to pass touch data through event.*
- struct [gslc\\_tsFont](#)  
*Font reference structure.*
- struct [gslc\\_tsImgRef](#)  
*Image reference structure.*
- struct [gslc\\_tsElemRef](#)

- Element reference structure.*
- struct [gslc\\_tsElem](#)
- Element Struct.*
- struct [gslc\\_tsCollect](#)
- Element collection struct.*
- struct [gslc\\_tsPage](#)
- Page structure.*
- struct [gslc\\_tsInputMap](#)
- Input mapping.*
- struct [gslc\\_tsGui](#)
- GUI structure.*

## Macros

- #define [GSLC\\_PMEM](#)
- #define [GSLC\\_FEATURE\\_FOCUS\\_ON\\_TOUCH](#)
- #define [GSLC\\_2PI](#)
- #define [GSLC\\_ELEM\\_FEA\\_NOSHRINK](#)
- Element features type.*
- #define [GSLC\\_ELEM\\_FEA\\_VALID](#)
- Element record is valid.*
- #define [GSLC\\_ELEM\\_FEA\\_FOCUS\\_EN](#)
- Element can accept focus.*
- #define [GSLC\\_ELEM\\_FEA\\_EDIT\\_EN](#)
- Element supports edit.*
- #define [GSLC\\_ELEM\\_FEA\\_ROUND\\_EN](#)
- Element is drawn with a rounded profile.*
- #define [GSLC\\_ELEM\\_FEA\\_CLICK\\_EN](#)
- Element accepts touch presses.*
- #define [GSLC\\_ELEM\\_FEA\\_GLOW\\_EN](#)
- Element supports glowing state.*
- #define [GSLC\\_ELEM\\_FEA\\_FRAME\\_EN](#)
- Element is drawn with a frame.*
- #define [GSLC\\_ELEM\\_FEA\\_FILL\\_EN](#)
- Element is drawn with a fill.*
- #define [GSLC\\_ELEM\\_FEA\\_NONE](#)
- Element default (no features set))*
- #define [GSLC\\_ALIGNV\\_TOP](#)
- Element text alignment.*
- #define [GSLC\\_ALIGNV\\_MID](#)
- Vertical align to middle.*
- #define [GSLC\\_ALIGNV\\_BOT](#)
- Vertical align to bottom.*
- #define [GSLC\\_ALIGNH\\_LEFT](#)
- Horizontal align to left.*
- #define [GSLC\\_ALIGNH\\_MID](#)
- Horizontal align to middle.*
- #define [GSLC\\_ALIGNH\\_RIGHT](#)
- Horizontal align to right.*
- #define [GSLC\\_ALIGN\\_TOP\\_LEFT](#)

- *Align to top-left.*
- `#define GSLC_ALIGN_TOP_MID`
- *Align to middle of top.*
- `#define GSLC_ALIGN_TOP_RIGHT`
- *Align to top-right.*
- `#define GSLC_ALIGN_MID_LEFT`
- *Align to middle of left side.*
- `#define GSLC_ALIGN_MID_MID`
- *Align to center.*
- `#define GSLC_ALIGN_MID_RIGHT`
- *Align to middle of right side.*
- `#define GSLC_ALIGN_BOT_LEFT`
- *Align to bottom-left.*
- `#define GSLC_ALIGN_BOT_MID`
- *Align to middle of bottom.*
- `#define GSLC_ALIGN_BOT_RIGHT`
- *Align to bottom-right.*
- `#define GSLC_COL_RED_DK4`
- *Basic color definition.*
- `#define GSLC_COL_RED_DK3`
- *Red (dark3)*
- `#define GSLC_COL_RED_DK2`
- *Red (dark2)*
- `#define GSLC_COL_RED_DK1`
- *Red (dark1)*
- `#define GSLC_COL_RED`
- *Red.*
- `#define GSLC_COL_RED_LT1`
- *Red (light1)*
- `#define GSLC_COL_RED_LT2`
- *Red (light2)*
- `#define GSLC_COL_RED_LT3`
- *Red (light3)*
- `#define GSLC_COL_RED_LT4`
- *Red (light4)*
- `#define GSLC_COL_GREEN_DK4`
- *Green (dark4)*
- `#define GSLC_COL_GREEN_DK3`
- *Green (dark3)*
- `#define GSLC_COL_GREEN_DK2`
- *Green (dark2)*
- `#define GSLC_COL_GREEN_DK1`
- *Green (dark1)*
- `#define GSLC_COL_GREEN`
- *Green.*
- `#define GSLC_COL_GREEN_LT1`
- *Green (light1)*
- `#define GSLC_COL_GREEN_LT2`
- *Green (light2)*
- `#define GSLC_COL_GREEN_LT3`
- *Green (light3)*

- #define [GSLC\\_COL\\_GREEN\\_LT4](#)  
*Green (light4)*
- #define [GSLC\\_COL\\_BLUE\\_DK4](#)  
*Blue (dark4)*
- #define [GSLC\\_COL\\_BLUE\\_DK3](#)  
*Blue (dark3)*
- #define [GSLC\\_COL\\_BLUE\\_DK2](#)  
*Blue (dark2)*
- #define [GSLC\\_COL\\_BLUE\\_DK1](#)  
*Blue (dark1)*
- #define [GSLC\\_COL\\_BLUE](#)  
*Blue.*
- #define [GSLC\\_COL\\_BLUE\\_LT1](#)  
*Blue (light1)*
- #define [GSLC\\_COL\\_BLUE\\_LT2](#)  
*Blue (light2)*
- #define [GSLC\\_COL\\_BLUE\\_LT3](#)  
*Blue (light3)*
- #define [GSLC\\_COL\\_BLUE\\_LT4](#)  
*Blue (light4)*
- #define [GSLC\\_COL\\_BLACK](#)  
*Black.*
- #define [GSLC\\_COL\\_GRAY\\_DK4](#)  
*Gray (dark4)*
- #define [GSLC\\_COL\\_GRAY\\_DK3](#)  
*Gray (dark3)*
- #define [GSLC\\_COL\\_GRAY\\_DK2](#)  
*Gray (dark2)*
- #define [GSLC\\_COL\\_GRAY\\_DK1](#)  
*Gray (dark1)*
- #define [GSLC\\_COL\\_GRAY](#)  
*Gray.*
- #define [GSLC\\_COL\\_GRAY\\_LT1](#)  
*Gray (light1)*
- #define [GSLC\\_COL\\_GRAY\\_LT2](#)  
*Gray (light2)*
- #define [GSLC\\_COL\\_GRAY\\_LT3](#)  
*Gray (light3)*
- #define [GSLC\\_COL\\_GRAY\\_LT4](#)  
*Gray (light4)*
- #define [GSLC\\_COL\\_WHITE](#)  
*White.*
- #define [GSLC\\_COL\\_YELLOW](#)  
*Yellow.*
- #define [GSLC\\_COL\\_YELLOW\\_DK](#)  
*Yellow (dark)*
- #define [GSLC\\_COL\\_PURPLE](#)  
*Purple.*
- #define [GSLC\\_COL\\_CYAN](#)  
*Cyan.*
- #define [GSLC\\_COL\\_MAGENTA](#)

- Magenta.*
  - #define `GSLC_COL_TEAL`
  - Teal.*
  - #define `GSLC_COL_ORANGE`
  - Orange.*
  - #define `GSLC_COL_BROWN`
  - Brown.*
  - #define `GSLC_COLMONO_BLACK`
  - Black.*
  - #define `GSLC_COLMONO_WHITE`
  - White.*
  - #define `TOUCH_ROTATION_DATA`
- Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.*
  - #define `TOUCH_ROTATION_SWAPXY(rotation)`
  - #define `TOUCH_ROTATION_FLIPX(rotation)`
  - #define `TOUCH_ROTATION_FLIPY(rotation)`
  - #define `GSLC_ELEMREF_DEFAULT`
- Define the default element reference flags for new elements.*
  - #define `GSLC_MIN(a, b)`
  - #define `GSLC_MAX(a, b)`
  - #define `TOUCH_ROTATION_DATA`
- Additional definitions for Touch Handling These macros define the transforms used in remapping the touchscreen inputs on the basis of the GUI nRotation setting.*
  - #define `TOUCH_ROTATION_SWAPXY(rotation)`
  - #define `TOUCH_ROTATION_FLIPX(rotation)`
  - #define `TOUCH_ROTATION_FLIPY(rotation)`
  - #define `GSLC_DEBUG_PRINT(sFmt, ...)`
- Macro to enable optional debug output.*
  - #define `GSLC_DEBUG2_PRINT(sFmt, ...)`
  - #define `GSLC_DEBUG_PRINT_CONST(sFmt, ...)`
  - #define `GSLC_DEBUG2_PRINT_CONST(sFmt, ...)`
  - #define `gslc_ElemCreateTxt_P(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, pFont, colTxt, colFrame, colFill, nAlignTxt, bFrameEn, bFillEn)`
- Create a read-only text element.*
  - #define `gslc_ElemCreateTxt_P_R(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, strLength, pFont, colTxt, colFrame, colFill, nAlignTxt, bFrameEn, bFillEn)`
- Create a read-write text element (element in Flash, string in RAM)*
  - #define `gslc_ElemCreateTxt_P_R_ext(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, strLength, pFont, colTxt, colTxtGlow, colFrame, colFill, nAlignTxt, nMarginX, nMarginY, bFrameEn, bFillEn, bClickEn, bGlowEn, pfuncXEvent, pfuncXDraw, pfuncXTouch, pfuncXTick)`
- Create a read-write text element (element in Flash, string in RAM) with extended customization options.*
  - #define `gslc_ElemCreateBox_P(pGui, nElemId, nPage, nX, nY, nW, nH, colFrame, colFill, bFrameEn, bFillEn, pfuncXDraw, pfuncXTick)`
- Create a read-only box element.*
  - #define `gslc_ElemCreateLine_P(pGui, nElemId, nPage, nX0, nY0, nX1, nY1, colFill)`
- Create a read-only line element.*
  - #define `gslc_ElemCreateBtnTxt_P(pGui, nElemId, nPage, nX, nY, nW, nH, strTxt, pFont, colTxt, colFrame, colFill, colFrameGlow, colFillGlow, nAlignTxt, bFrameEn, bFillEn, callFunc, extraData)`
- Create a text button element.*

## Typedefs

- typedef int16\_t(\* [GSLC\\_CB\\_DEBUG\\_OUT](#)) (char ch)
- typedef struct [gslc\\_tsElem](#) [gslc\\_tsElem](#)  
*Element Struct.*
- typedef struct [gslc\\_tsEvent](#) [gslc\\_tsEvent](#)  
*Event structure.*
- typedef bool(\* [GSLC\\_CB\\_EVENT](#)) (void \*pvGui, [gslc\\_tsEvent](#) sEvent)  
*Callback function for element drawing.*
- typedef bool(\* [GSLC\\_CB\\_DRAW](#)) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Callback function for element drawing.*
- typedef bool(\* [GSLC\\_CB\\_TOUCH](#)) (void \*pvGui, void \*pvElemRef, [gslc\\_teTouch](#) eTouch, int16\_t nX, int16\_t nY)  
*Callback function for element touch tracking.*
- typedef bool(\* [GSLC\\_CB\\_TICK](#)) (void \*pvGui, void \*pvElemRef)  
*Callback function for element tick.*
- typedef bool(\* [GSLC\\_CB\\_PIN\\_POLL](#)) (void \*pvGui, int16\_t \*pnPinInd, int16\_t \*pnPinVal)  
*Callback function for pin polling.*
- typedef bool(\* [GSLC\\_CB\\_INPUT](#)) (void \*pvGui, void \*pvElemRef, int16\_t nStatus, void \*pvData)  
*Callback function for element input ready.*
- typedef struct [gslc\\_tsRect](#) [gslc\\_tsRect](#)  
*Rectangular region. Defines X,Y corner coordinates plus dimensions.*
- typedef struct [gslc\\_tsPt](#) [gslc\\_tsPt](#)  
*Define point coordinates.*
- typedef struct [gslc\\_tsColor](#) [gslc\\_tsColor](#)  
*Color structure. Defines RGB triplet.*
- typedef struct [gslc\\_tsRectState](#) [gslc\\_tsRectState](#)  
*State associated with an element's region.*
- typedef struct [gslc\\_tsEventTouch](#) [gslc\\_tsEventTouch](#)  
*Structure used to pass touch data through event.*

## Enumerations

- enum [gslc\\_teElemId](#) {  
    [GSLC\\_ID\\_USER\\_BASE](#), [GSLC\\_ID\\_NONE](#), [GSLC\\_ID\\_AUTO](#), [GSLC\\_ID\\_TEMP](#),  
    [GSLC\\_ID\\_AUTO\\_BASE](#) }  
*Element ID enumerations.*
- enum [gslc\\_tePageId](#) { [GSLC\\_PAGE\\_USER\\_BASE](#), [GSLC\\_PAGE\\_NONE](#) }  
*Page ID enumerations.*
- enum [gslc\\_teStackPage](#) { [GSLC\\_STACK\\_BASE](#), [GSLC\\_STACK\\_CUR](#), [GSLC\\_STACK\\_OVERLAY](#), [GSLC\\_STACK\\_MAX](#) }  
*Define page stack.*
- enum [gslc\\_teGroupId](#) { [GSLC\\_GROUP\\_ID\\_USER\\_BASE](#), [GSLC\\_GROUP\\_ID\\_NONE](#) }  
*Group ID enumerations.*
- enum [gslc\\_teFontId](#) { [GSLC\\_FONT\\_USER\\_BASE](#), [GSLC\\_FONT\\_NONE](#) }  
*Font ID enumerations.*
- enum [gslc\\_teElemInd](#) { [GSLC\\_IND\\_NONE](#), [GSLC\\_IND\\_FIRST](#) }  
*Element Index enumerations.*
- enum [gslc\\_teTypeCore](#) {  
    [GSLC\\_TYPE\\_NONE](#), [GSLC\\_TYPE\\_BKGND](#), [GSLC\\_TYPE\\_BTN](#), [GSLC\\_TYPE\\_TXT](#),  
    [GSLC\\_TYPE\\_BOX](#), [GSLC\\_TYPE\\_LINE](#), [GSLC\\_TYPE\\_BASE\\_EXTEND](#) }  
*Element Type enumerations.*

*Element type.*

- enum `gslc_teInputRawEvent` {  
`GSLC_INPUT_NONE`, `GSLC_INPUT_TOUCH`, `GSLC_INPUT_KEY_DOWN`, `GSLC_INPUT_KEY_UP`,  
`GSLC_INPUT_PIN_ASSERT`, `GSLC_INPUT_PIN_DEASSERT` }

*Raw input event types: touch, key, GPIOs.*

- enum `gslc_teInputMode` { `GSLC_INPUTMODE_NAV`, `GSLC_INPUTMODE_EDIT` }

*External input mode.*

- enum `gslc_teAction` {  
`GSLC_ACTION_UNDEF`, `GSLC_ACTION_NONE`, `GSLC_ACTION_FOCUS_PREV`, `GSLC_ACTION_FO↵CUS_NEXT`,  
`GSLC_ACTION_PRESELECT`, `GSLC_ACTION_SELECT`, `GSLC_ACTION_SET_REL`, `GSLC_ACTION_↵SET_ABS`,  
`GSLC_ACTION_DEBUG` }

*GUI Action Requested These actions are usually the result of an InputMap lookup.*

- enum `gslc_tePin` {  
`GSLC_PIN_BTN_A`, `GSLC_PIN_BTN_A_LONG`, `GSLC_PIN_BTN_B`, `GSLC_PIN_BTN_B_LONG`,  
`GSLC_PIN_BTN_C`, `GSLC_PIN_BTN_C_LONG`, `GSLC_PIN_BTN_D`, `GSLC_PIN_BTN_D_LONG`,  
`GSLC_PIN_BTN_E`, `GSLC_PIN_BTN_E_LONG`, `GSLC_PIN_BTN_UP`, `GSLC_PIN_BTN_DOWN`,  
`GSLC_PIN_BTN_LEFT`, `GSLC_PIN_BTN_RIGHT`, `GSLC_PIN_BTN_SEL` }

*General purpose pin/button constants.*

- enum `gslc_teTouch` {  
`GSLC_TOUCH_NONE`, `GSLC_TOUCH_TYPE_MASK`, `GSLC_TOUCH_COORD`, `GSLC_TOUCH_DIRECT`,  
`GSLC_TOUCH_SUBTYPE_MASK`, `GSLC_TOUCH_DOWN`, `GSLC_TOUCH_DOWN_IN`, `GSLC_TOUCH_↵_DOWN_OUT`,  
`GSLC_TOUCH_UP`, `GSLC_TOUCH_UP_IN`, `GSLC_TOUCH_UP_OUT`, `GSLC_TOUCH_MOVE`,  
`GSLC_TOUCH_MOVE_IN`, `GSLC_TOUCH_MOVE_OUT`, `GSLC_TOUCH_FOCUS_ON`, `GSLC_TOUCH_↵_FOCUS_OFF`,  
`GSLC_TOUCH_FOCUS_PRESELECT`, `GSLC_TOUCH_FOCUS_SELECT`, `GSLC_TOUCH_SET_REL`,  
`GSLC_TOUCH_SET_ABS` }

*Processed event from input raw events and actions.*

- enum `gslc_telnitStat` { `GSLC_INITSTAT_UNDEF`, `GSLC_INITSTAT_INACTIVE`, `GSLC_INITSTAT_FAIL`,  
`GSLC_INITSTAT_ACTIVE` }

*Status of a module's initialization.*

- enum `gslc_teEventType` {  
`GSLC_EVT_NONE`, `GSLC_EVT_DRAW`, `GSLC_EVT_TOUCH`, `GSLC_EVT_TICK`,  
`GSLV_EVT_CUSTOM` }

*Event types.*

- enum `gslc_teEventSubType` { `GSLC_EVTSUB_NONE`, `GSLC_EVTSUB_DRAW_NEEDED`, `GSLC_EVTS_↵UB_DRAW_FORCE` }

*Event sub-types.*

- enum `gslc_teRedrawType` { `GSLC_REDRAW_NONE`, `GSLC_REDRAW_FULL`, `GSLC_REDRAW_INC`, `G_↵SLC_REDRAW_FOCUS` }

*Redraw types.*

- enum `gslc_teFontRefType` { `GSLC_FONTREF_FNAME`, `GSLC_FONTREF_PTR` }

*Font Reference types.*

- enum `gslc_teFontRefMode` { `GSLC_FONTREF_MODE_DEFAULT`, `GSLC_FONTREF_MODE_1`, `GSLC_↵FONTREF_MODE_2`, `GSLC_FONTREF_MODE_3` }

*Font Reference modes.*

- enum `gslc_teElemRefFlags` {  
`GSLC_ELEMREF_NONE`, `GSLC_ELEMREF_SRC_RAM`, `GSLC_ELEMREF_SRC_PROG`, `GSLC_ELEM_↵REF_SRC_CONST`,  
`GSLC_ELEMREF_REDRAW_NONE`, `GSLC_ELEMREF_REDRAW_FULL`, `GSLC_ELEMREF_REDRAW_↵_INC`, `GSLC_ELEMREF_REDRAW_FOCUS`,  
`GSLC_ELEMREF_EDITING`, `GSLC_ELEMREF_FOCUSED`, `GSLC_ELEMREF_GLOWING`, `GSLC_ELE_↵MREF_VISIBLE`,  
`GSLC_ELEMREF_SRC`, `GSLC_ELEMREF_REDRAW_MASK` }



*Element reference flags: Describes characteristics of an element.*

- enum [gslc\\_telmRefFlags](#) {  
[GSLC\\_IMGREF\\_NONE](#), [GSLC\\_IMGREF\\_SRC\\_FILE](#), [GSLC\\_IMGREF\\_SRC\\_SD](#), [GSLC\\_IMGREF\\_SRC\\_RAM](#),  
[GSLC\\_IMGREF\\_SRC\\_PROG](#), [GSLC\\_IMGREF\\_FMT\\_BMP24](#), [GSLC\\_IMGREF\\_FMT\\_BMP16](#), [GSLC\\_IMGREF\\_FMT\\_RAW1](#),  
[GSLC\\_IMGREF\\_FMT\\_JPG](#), [GSLC\\_IMGREF\\_SRC](#), [GSLC\\_IMGREF\\_FMT](#) }

*Image reference flags: Describes characteristics of an image reference.*

- enum [gslc\\_teTxtFlags](#) {  
[GSLC\\_TXT\\_MEM\\_RAM](#), [GSLC\\_TXT\\_MEM\\_PROG](#), [GSLC\\_TXT\\_ALLOC\\_NONE](#), [GSLC\\_TXT\\_ALLOC\\_INT](#),  
[GSLC\\_TXT\\_ALLOC\\_EXT](#), [GSLC\\_TXT\\_ENC\\_PLAIN](#), [GSLC\\_TXT\\_ENC\\_UTF8](#), [GSLC\\_TXT\\_MEM](#),  
[GSLC\\_TXT\\_ALLOC](#), [GSLC\\_TXT\\_ENC](#), [GSLC\\_TXT\\_DEFAULT](#) }

*Text reference flags: Describes the characteristics of a text string (ie.*

## Functions

- char \* [gslc\\_GetVer](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice version number.*
- const char \* [gslc\\_GetNameDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice display driver name.*
- const char \* [gslc\\_GetNameTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the GUISlice touch driver name.*
- void \* [gslc\\_GetDriverDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native display driver instance.*
- void \* [gslc\\_GetDriverTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native touch driver instance.*
- bool [gslc\\_Init](#) ([gslc\\_tsGui](#) \*pGui, void \*pvDriver, [gslc\\_tsPage](#) \*asPage, uint8\_t nMaxPage, [gslc\\_tsFont](#) \*asFont, uint8\_t nMaxFont)  
*Initialize the GUISlice library.*
- void [gslc\\_InitDebug](#) ([GSLC\\_CB\\_DEBUG\\_OUT](#) pfunc)  
*Initialize debug output.*
- void [gslc\\_DebugPrintf](#) (const char \*pFmt,...)  
*Optimized printf routine for GUISlice debug/error output.*
- bool [gslc\\_GuiRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)  
*Dynamically change rotation, automatically adapt touchscreen axes swap/flip.*
- void [gslc\\_Quit](#) ([gslc\\_tsGui](#) \*pGui)  
*Exit the GUISlice environment.*
- void [gslc\\_Update](#) ([gslc\\_tsGui](#) \*pGui)  
*Perform main GUISlice handling functions.*
- bool [gslc\\_SetBkgndImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tslmgRef](#) slmgRef)  
*Configure the background to use a bitmap image.*
- bool [gslc\\_SetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the background to use a solid color.*
- bool [gslc\\_SetTransparentColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the color to use for image transparency.*
- [gslc\\_tsRect](#) [gslc\\_GetClipRect](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the current the clipping rectangle.*
- bool [gslc\\_SetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for further drawing.*
- bool [gslc\\_IsInRect](#) (int16\_t nSelX, int16\_t nSelY, [gslc\\_tsRect](#) rRect)  
*Determine if a coordinate is inside of a rectangular region.*

- [gslc\\_tsRect gslc\\_ExpandRect](#) ([gslc\\_tsRect](#) rRect, [int16\\_t](#) nExpandW, [int16\\_t](#) nExpandH)  
*Expand or contract a rectangle in width and/or height (equal amounts on both side), based on the centerpoint of the rectangle.*
- [bool gslc\\_IsInWH](#) ([int16\\_t](#) nSelX, [int16\\_t](#) nSelY, [uint16\\_t](#) nWidth, [uint16\\_t](#) nHeight)  
*Determine if a coordinate is inside of a width x height region.*
- [void gslc\\_UnionRect](#) ([gslc\\_tsRect](#) \*pRect, [gslc\\_tsRect](#) rAddRect)  
*Expand a rect to include another rect.*
- [void gslc\\_InvalidateRgnReset](#) ([gslc\\_tsGui](#) \*pGui)  
*Reset the invalidation region.*
- [void gslc\\_InvalidateRgnPage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage)  
*Include an entire page (eg.*
- [void gslc\\_InvalidateRgnScreen](#) ([gslc\\_tsGui](#) \*pGui)  
*Mark the entire screen as invalidated.*
- [void gslc\\_InvalidateRgnAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rAddRect)  
*Add a rectangular region to the invalidation region.*
- [bool gslc\\_ClipPt](#) ([gslc\\_tsRect](#) \*pClipRect, [int16\\_t](#) nX, [int16\\_t](#) nY)  
*Perform basic clipping of a single point to a clipping region.*
- [bool gslc\\_ClipLine](#) ([gslc\\_tsRect](#) \*pClipRect, [int16\\_t](#) \*pnX0, [int16\\_t](#) \*pnY0, [int16\\_t](#) \*pnX1, [int16\\_t](#) \*pnY1)  
*Perform basic clipping of a line to a clipping region.*
- [bool gslc\\_ClipRect](#) ([gslc\\_tsRect](#) \*pClipRect, [gslc\\_tsRect](#) \*pRect)  
*Perform basic clipping of a rectangle to a clipping region.*
- [gslc\\_tslmgRef gslc\\_GetImageFromFile](#) (const char \*pFname, [gslc\\_telmgRefFlags](#) eFmt)  
*Create an image reference to a bitmap file in LINUX filesystem.*
- [gslc\\_tslmgRef gslc\\_GetImageFromSD](#) (const char \*pFname, [gslc\\_telmgRefFlags](#) eFmt)  
*Create an image reference to a bitmap file in SD card.*
- [gslc\\_tslmgRef gslc\\_GetImageFromRam](#) (unsigned char \*plmgBuf, [gslc\\_telmgRefFlags](#) eFmt)  
*Create an image reference to a bitmap in SRAM.*
- [gslc\\_tslmgRef gslc\\_GetImageFromProg](#) (const unsigned char \*plmgBuf, [gslc\\_telmgRefFlags](#) eFmt)  
*Create an image reference to a bitmap in program memory (PROGMEM)*
- [void gslc\\_PolarToXY](#) ([uint16\\_t](#) nRad, [int16\\_t](#) n64Ang, [int16\\_t](#) \*nDX, [int16\\_t](#) \*nDY)  
*Convert polar coordinate to cartesian.*
- [int16\\_t gslc\\_sinFX](#) ([int16\\_t](#) n64Ang)  
*Calculate fixed-point sine function from fractional degrees.*
- [int16\\_t gslc\\_cosFX](#) ([int16\\_t](#) n64Ang)  
*Calculate fixed-point cosine function from fractional degrees.*
- [gslc\\_tsColor gslc\\_ColorBlend2](#) ([gslc\\_tsColor](#) colStart, [gslc\\_tsColor](#) colEnd, [uint16\\_t](#) nMidAmt, [uint16\\_t](#) n↔BlendAmt)  
*Create a color based on a blend between two colors.*
- [gslc\\_tsColor gslc\\_ColorBlend3](#) ([gslc\\_tsColor](#) colStart, [gslc\\_tsColor](#) colMid, [gslc\\_tsColor](#) colEnd, [uint16\\_t](#) n↔MidAmt, [uint16\\_t](#) nBlendAmt)  
*Create a color based on a blend between three colors.*
- [bool gslc\\_ColorEqual](#) ([gslc\\_tsColor](#) a, [gslc\\_tsColor](#) b)  
*Check whether two colors are equal.*
- [void gslc\\_DrawSetPixel](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nX, [int16\\_t](#) nY, [gslc\\_tsColor](#) nCol)  
*Set a pixel on the active screen to the given color with lock.*
- [void gslc\\_DrawLine](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nX0, [int16\\_t](#) nY0, [int16\\_t](#) nX1, [int16\\_t](#) nY1, [gslc\\_tsColor](#) nCol)  
*Draw an arbitrary line using Bresenham's algorithm.*
- [void gslc\\_DrawLineH](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nX, [int16\\_t](#) nY, [uint16\\_t](#) nW, [gslc\\_tsColor](#) nCol)  
*Draw a horizontal line.*
- [void gslc\\_DrawLineV](#) ([gslc\\_tsGui](#) \*pGui, [int16\\_t](#) nX, [int16\\_t](#) nY, [uint16\\_t](#) nH, [gslc\\_tsColor](#) nCol)  
*Draw a vertical line.*

- void [gslc\\_DrawLinePolar](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, uint16\_t nRadStart, uint16\_t nRadEnd, int16\_t n64Ang, [gslc\\_tsColor](#) nCol)  
*Draw a polar ray segment.*
- void [gslc\\_DrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a framed rectangle.*
- void [gslc\\_DrawFrameRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)  
*Draw a framed rounded rectangle.*
- void [gslc\\_DrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a filled rectangle.*
- void [gslc\\_DrawFillRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)  
*Draw a filled rounded rectangle.*
- void [gslc\\_DrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)  
*Draw a framed circle.*
- void [gslc\\_DrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)  
*Draw a filled circle.*
- void [gslc\\_DrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)  
*Draw a framed triangle.*
- void [gslc\\_DrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)  
*Draw a filled triangle.*
- void [gslc\\_DrawFrameQuad](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*psPt, [gslc\\_tsColor](#) nCol)  
*Draw a framed quadrilateral.*
- void [gslc\\_DrawFillQuad](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*psPt, [gslc\\_tsColor](#) nCol)  
*Draw a filled quadrilateral.*
- void [gslc\\_DrawFillGradSector](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArcStart, [gslc\\_tsColor](#) cArcEnd, int16\_t nAngSecStart, int16\_t nAngSecEnd, int16\_t nAngGradStart, int16\_t nAngGradRange)  
*Draw a gradient filled sector of a circle with support for inner and outer radius.*
- void [gslc\\_DrawFillSector](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nQuality, int16\_t nMidX, int16\_t nMidY, int16\_t nRad1, int16\_t nRad2, [gslc\\_tsColor](#) cArc, int16\_t nAngSecStart, int16\_t nAngSecEnd)  
*Draw a flat filled sector of a circle with support for inner and outer radius.*
- bool [gslc\\_FontAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)  
*Load a font into the local font cache and assign font ID (nFontId).*
- bool [gslc\\_FontSet](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)  
*Load a font into the local font cache and store as font ID (nFontId)*
- [gslc\\_tsFont](#) \* [gslc\\_FontGet](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId)  
*Fetch a font from its ID value.*
- bool [gslc\\_FontSetMode](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nFontId, [gslc\\_teFontRefMode](#) eFontMode)  
*Set the font operating mode.*
- int [gslc\\_GetPageCur](#) ([gslc\\_tsGui](#) \*pGui)  
*Fetch the current page ID.*
- void [gslc\\_SetStackPage](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nStackPos, int16\_t nPageId)  
*Assign a page to the page stack.*
- void [gslc\\_SetStackState](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nStackPos, bool bActive, bool bDoDraw)  
*Change the status of a page in a page stack.*
- void [gslc\\_SetPageBase](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId)  
*Assigns a page for the base layer in the page stack.*

- void [gslc\\_SetPageCur](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId)  
*Select a page for the current layer in the page stack.*
- void [gslc\\_SetPageOverlay](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId)  
*Select a page for the overlay layer in the page stack.*
- void [gslc\\_PopupShow](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, bool bModal)  
*Show a popup dialog.*
- void [gslc\\_PopupHide](#) ([gslc\\_tsGui](#) \*pGui)  
*Hides the currently active popup dialog.*
- void [gslc\\_PageRedrawSet](#) ([gslc\\_tsGui](#) \*pGui, bool bRedraw)  
*Update the need-redraw status for the current page.*
- bool [gslc\\_PageRedrawGet](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the need-redraw status for the current page.*
- void [gslc\\_PageAdd](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, [gslc\\_tsElem](#) \*psElem, uint16\_t nMaxElem, [gslc\\_tsElemRef](#) \*psElemRef, uint16\_t nMaxElemRef)  
*Add a page to the GUI.*
- [gslc\\_tsElemRef](#) \* [gslc\\_PageFindElemById](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, int16\_t nElemId)  
*Find an element in the GUI by its Page ID and Element ID.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId)  
*Create a Text Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, char \*pStrBuf, uint8\_t nStrBufMax, int16\_t nFontId, [GSLC\\_CB\\_TOUCH](#) cbTouch)  
*Create a textual Button Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBtnImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, [gslc\\_slmgRef](#) slmgRef, [gslc\\_slmgRef](#) slmgRefSel, [GSLC\\_CB\\_TOUCH](#) cbTouch)  
*Create a graphical Button Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateBox](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem)  
*Create a Box Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1)  
*Create a Line Element.*
- [gslc\\_tsElemRef](#) \* [gslc\\_ElemCreateImg](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nElemId, int16\_t nPage, [gslc\\_tsRect](#) rElem, [gslc\\_slmgRef](#) slmgRef)  
*Create an image Element.*
- int [gslc\\_ElemGetId](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get an Element ID from an element structure.*
- void [gslc\\_ElemSetFillEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFillEn)  
*Set the fill state for an Element.*
- void [gslc\\_ElemSetFrameEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFrameEn)  
*Set the frame state for an Element.*
- void [gslc\\_ElemSetRoundEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bRoundEn)  
*Set the rounded frame/fill state for an Element.*
- void [gslc\\_ElemSetCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colFrame, [gslc\\_tsColor](#) colFill, [gslc\\_tsColor](#) colFillGlow)  
*Update the common color selection for an Element.*
- void [gslc\\_ElemSetGlowCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colFrameGlow, [gslc\\_tsColor](#) colFillGlow, [gslc\\_tsColor](#) colTxtGlow)  
*Update the common color selection for glowing state of an Element.*
- void [gslc\\_ElemSetGroup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int nGroupId)  
*Set the group ID for an element.*
- int [gslc\\_ElemGetGroup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)

- Get the group ID for an element.*
- void [gslc\\_ElemSetRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRect](#) rElem)
- Set the position and size for an element.*
- [gslc\\_tsRect](#) [gslc\\_ElemGetRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- Get the rectangular region for an element.*
- void [gslc\\_ElemSetTxtAlign](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, unsigned nAlign)
- Set the alignment of a textual element (horizontal and vertical)*
- void [gslc\\_ElemSetTxtMargin](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, unsigned nMargin)
- Set the margin around of a textual element.*
- void [gslc\\_ElemSetTxtMarginXY](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nMarginX, int8\_t nMarginY)
- Set the margin around of a textual element (X & Y offsets can be different)*
- void [gslc\\_StrCopy](#) (char \*pDstStr, const char \*pSrcStr, uint16\_t nDstLen)
- Helper routine to perform string deep copy.*
- void [gslc\\_ElemSetTxtStr](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, const char \*pStr)
- Update the text string associated with an Element.*
- char \* [gslc\\_ElemGetTxtStr](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- Fetch the current text string associated with an Element.*
- void [gslc\\_ElemSetTxtCol](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsColor](#) colVal)
- Update the text string color associated with an Element ID.*
- void [gslc\\_ElemSetTxtMem](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)
- Update the text string location in memory.*
- void [gslc\\_ElemSetTxtEnc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teTxtFlags](#) eFlags)
- Update the text string encoding mode.*
- void [gslc\\_ElemUpdateFont](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int nFontId)
- Update the Font selected for an Element's text.*
- void [gslc\\_ElemSetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)
- Update the need-redraw status for an element.*
- [gslc\\_teRedrawType](#) [gslc\\_ElemGetRedraw](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- Get the need-redraw status for an element.*
- void [gslc\\_ElemSetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowEn)
- Update the glowing enable for an element.*
- void [gslc\\_ElemSetClickEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bClickEn)
- Update the click enable for an element.*
- void [gslc\\_ElemSetTouchFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_TOUCH](#) funcCb)
- Update the touch function callback for an element.*
- void [gslc\\_ElemSetStyleFrom](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRefSrc, [gslc\\_tsElemRef](#) \*pElemRefDest)
- Copy style settings from one element to another.*
- void [gslc\\_ResetRectState](#) ([gslc\\_tsRectState](#) \*pState)
- Reset the element region state struct.*
- void [gslc\\_ElemCalcRectState](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsRectState](#) \*pState)
- Calculate the element region state struct.*
- int8\_t [gslc\\_ElemCalcResizeForFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- Calculate the change in dimensions of an element to account for any change in focus and/or frame attributes.*
- void [gslc\\_ElemGrowRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int8\_t nDelta)
- Increase or decrease the size of an element's region.*
- bool [gslc\\_ElemGetGlowEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)
- Get the glowing enable for an element.*
- void [gslc\\_ElemSetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bGlowing)
- Update the glowing indicator for an element.*

- bool [gslc\\_ElemGetGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the glowing indicator for an element.*
- bool [gslc\\_ElemGetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the focus enable for an element.*
- void [gslc\\_ElemSetFocusEn](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocusEn)  
*Set the focus enable for an element.*
- void [gslc\\_ElemSetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bFocused)  
*Update the focused indicator for an element.*
- bool [gslc\\_ElemGetFocus](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the focused indicator for an element.*
- void [gslc\\_ElemSetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bEditing)  
*Update the editing indicator for an element.*
- bool [gslc\\_ElemGetEdit](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the editing indicator for an element.*
- void [gslc\\_ElemSetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, bool bVisible)  
*Update the visibility status for an element.*
- bool [gslc\\_ElemGetVisible](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Get the visibility status for an element.*
- bool [gslc\\_ElemGetOnScreen](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef)  
*Determine whether an element is visible on the screen.*
- void [gslc\\_ElemSetDrawFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_DRAW](#) funcCb)  
*Assign the drawing callback function for an element.*
- void [gslc\\_ElemSetTickFunc](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [GSLC\\_CB\\_TICK](#) funcCb)  
*Assign the tick callback function for an element.*
- bool [gslc\\_ElemOwnsCoord](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, int16\_t nX, int16\_t nY, bool b↵  
OnlyClickEn)  
*Determine if a coordinate is inside of an element.*
- bool [gslc\\_InitTouch](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)  
*Initialize the touchscreen device driver.*
- bool [gslc\\_GetTouch](#) ([gslc\\_tsGui](#) \*pGui, int16\_t \*pnX, int16\_t \*pnY, uint16\_t \*pnPress, [gslc\\_telInputRawEvent](#) \*peInputEvent, int16\_t \*pnInputVal)  
*Initialize the touchscreen device driver.*
- void [gslc\\_SetTouchRemapEn](#) ([gslc\\_tsGui](#) \*pGui, bool bEn)  
*Configure touchscreen remapping.*
- void [gslc\\_SetTouchRemapCal](#) ([gslc\\_tsGui](#) \*pGui, uint16\_t nXMin, uint16\_t nXMax, uint16\_t nYMin, uint16\_t nYMax)  
*Configure touchscreen calibration remapping values.*
- void [gslc\\_SetTouchPressCal](#) ([gslc\\_tsGui](#) \*pGui, uint16\_t nPressMin, uint16\_t nPressMax)  
*Configure touchscreen calibration pressure values.*
- void [gslc\\_SetTouchRemapYX](#) ([gslc\\_tsGui](#) \*pGui, bool bSwap)  
*Configure touchscreen XY swap.*
- void [gslc\\_SetTouchEn](#) ([gslc\\_tsGui](#) \*pGui, bool bEn)  
*Make touchscreen sensitive (GUI reacts to touch events) or insensitive (GUI ignores touch events)*
- bool [gslc\\_GetTouchEn](#) ([gslc\\_tsGui](#) \*pGui)  
*Get whether the GUI will react to touch events or not.*
- void [gslc\\_SetPinPollFunc](#) ([gslc\\_tsGui](#) \*pGui, [GSLC\\_CB\\_PIN\\_POLL](#) pfunc)  
*Specify the callback function that is used to collect the state of any external inputs (eg.*
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)  
*Specify the mapping between external pin inputs (fetched by the SetPinPollFunc() callback and the GUI actions.*
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_te↵  
Action](#) eAction, int16\_t nActionVal)



- Add an entry into the external input mapping table.*

  - `gslc_tsElemRef * gslc_FocusElemGet (gslc_tsGui *pGui)`
- Find the currently focused element.*

  - `void gslc_FocusPageStep (gslc_tsGui *pGui, bool bNext)`
- Advance the focus to the next page in the page stack.*

  - `int16_t gslc_FocusElemStep (gslc_tsGui *pGui, bool bNext)`
- Advance the focus to the next element in the focused page.*

  - `void gslc_FocusElemIndSet (gslc_tsGui *pGui, int16_t nPageInd, int16_t nElemInd, bool bFocus)`
- Change the focus to the indexed element on the specified page.*

  - `void gslc_FocusSetToTrackedElem (gslc_tsGui *pGui, gslc_tsCollect *pCollect)`
- Change the focus to the currently-tracked element.*

  - `gslc_tsImgRef gslc_ResetImage ()`
- Create a blank image reference structure.*

  - `gslc_tsElem gslc_ElemCreate (gslc_tsGui *pGui, int16_t nElemId, int16_t nPageId, int16_t nType, gslc_tsRect rElem, char *pStrBuf, uint8_t nStrBufMax, int16_t nFontId)`
- Create a new element with default styling.*

  - `gslc_tsElemRef * gslc_ElemAdd (gslc_tsGui *pGui, int16_t nPageId, gslc_tsElem *pElem, gslc_tsElemRefFlags eFlags)`
- Add the Element to the list of generated elements in the GUI environment.*

  - `uint8_t gslc_GetElemRefFlag (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nFlagMask)`
- Get the flags associated with an element reference.*

  - `void gslc_SetElemRefFlag (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nFlagMask, uint8_t nFlagVal)`
- Set the flags associated with an element reference.*

  - `gslc_tsElem * gslc_GetElemFromRef (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)`
- Returns a pointer to an element from an element reference, copying from FLASH to RAM if element is stored in PROGMEM.*

  - `gslc_tsElem * gslc_GetElemFromRefD (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nLineNum)`
- Returns a pointer to an element from an element reference.*

  - `void * gslc_GetXDataFromRef (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nType, int16_t nLineNum)`
- Returns a pointer to the data structure associated with an extended element.*

  - `void gslc_ElemSetImage (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsImgRef sImgRef, gslc_tsImgRef sImgRefSel)`
- Set an element to use a bitmap image.*

  - `bool gslc_ElemDrawByRef (gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsRedrawType eRedraw)`
- Draw an element to the active display.*

  - `void gslc_ElemDraw (gslc_tsGui *pGui, int16_t nPageId, int16_t nElemId)`
- Draw an element to the active display.*

  - `void gslc_DrawTxtBase (gslc_tsGui *pGui, char *pStrBuf, gslc_tsRect rTxt, gslc_tsFont *pTxtFont, gslc_tsTxtFlags eTxtFlags, int8_t eTxtAlign, gslc_tsColor colTxt, gslc_tsColor colBg, int16_t nMarginW, int16_t nMarginH)`
- Draw text with full text justification.*

  - `void gslc_SetRoundRadius (gslc_tsGui *pGui, uint8_t nRadius)`
- Set the global rounded radius.*

  - `void gslc_SetFocusCol (gslc_tsGui *pGui, gslc_tsColor colFocusNone, gslc_tsColor colFocus, gslc_tsColor colFocusEdit)`
- Set the global focus color choices.*

  - `bool gslc_PageEvent (void *pvGui, gslc_tsEvent sEvent)`
- Common event handler function for a page.*

  - `void gslc_PageRedrawGo (gslc_tsGui *pGui)`
- Redraw all elements on the active page.*

- void `gslc_PageFlipSet` (`gslc_tsGui` \*pGui, bool bNeeded)  
*Indicate whether the screen requires page flip.*
- bool `gslc_PageFlipGet` (`gslc_tsGui` \*pGui)  
*Get state of pending page flip state.*
- void `gslc_PageFlipGo` (`gslc_tsGui` \*pGui)  
*Update the visible screen if page has been marked for flipping.*
- `gslc_tsPage` \* `gslc_PageFindById` (`gslc_tsGui` \*pGui, int16\_t nPageId)  
*Find a page in the GUI by its ID.*
- void `gslc_PageRedrawCalc` (`gslc_tsGui` \*pGui)  
*Perform a redraw calculation on the page to determine if additional elements should also be redrawn.*
- `gslc_tsEvent` `gslc_EventCreate` (`gslc_tsGui` \*pGui, `gslc_teEventType` eType, uint8\_t nSubType, void \*pv↔ Scope, void \*pvData)  
*Create an event structure.*
- bool `gslc_ElemEvent` (void \*pvGui, `gslc_tsEvent` sEvent)  
*Common event handler function for an element.*
- bool `gslc_ElemSendEventTouch` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRefTracked, `gslc_teTouch` e↔ Touch, int16\_t nX, int16\_t nY)  
*Trigger an element's touch event.*
- void `gslc_CollectReset` (`gslc_tsCollect` \*pCollect, `gslc_tsElem` \*asElem, uint16\_t nElemMax, `gslc_tsElemRef` \*asElemRef, uint16\_t nElemRefMax)  
*Reset the members of an element collection.*
- `gslc_tsElemRef` \* `gslc_CollectElemAdd` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, const `gslc_tsElem` \*p↔ Elem, `gslc_teElemRefFlags` eFlags)  
*Add an element to a collection.*
- bool `gslc_CollectGetRedraw` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Determine if any elements in a collection need redraw.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemById` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, int16\_t nElemId)  
*Find an element in a collection by its Element ID.*
- `gslc_tsElemRef` \* `gslc_CollectFindElemFromCoord` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, int16\_t nX, int16\_t nY, int16\_t \*pnElemInd)  
*Find an element in a collection by a coordinate coordinate.*
- int `gslc_CollectGetNextId` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Allocate the next available Element ID in a collection.*
- `gslc_tsElemRef` \* `gslc_CollectGetElemRefTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Get the element within a collection that is currently being tracked.*
- void `gslc_CollectSetElemTracked` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRef, int16\_t nElemInd)  
*Set the element within a collection that is currently being tracked.*
- void `gslc_CollectSetParent` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRefParent)  
*Assign the parent element reference to all elements within a collection.*
- bool `gslc_CollectEvent` (void \*pvGui, `gslc_tsEvent` sEvent)  
*Common event handler function for an element collection.*
- void `gslc_CollectTouch` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsEventTouch` \*pEventTouch)  
*Handle touch events within the element collection.*
- bool `gslc_CollectTouchCompound` (void \*pvGui, void \*pvElemRef, `gslc_teTouch` eTouch, int16\_t nRelX, int16\_t nRelY, `gslc_tsCollect` \*pCollect)  
*Handle dispatch of touch (up,down,move) events to compound elements sub elements.*
- void `gslc_CollectInput` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsEventTouch` \*pEventTouch)  
*Handle direct input events within the element collection.*
- void `gslc_TrackTouch` (`gslc_tsGui` \*pGui, `gslc_tsPage` \*pPage, int16\_t nX, int16\_t nY, uint16\_t nPress)  
*Handles a touch event and performs the necessary tracking, glowing and selection actions depending on the press state.*



- void [gslc\\_TrackInput](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal)  
*Handles a direct input event and performs the necessary tracking, glowing and selection actions depending on the state.*
- bool [gslc\\_InputMapLookup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) \*peAction, int16\_t \*pnActionVal)
- void [gslc\\_GuiDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Free up any surfaces associated with the GUI, pages, collections and elements.*
- void [gslc\\_PageDestruct](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage)  
*Free up any members associated with a page.*
- void [gslc\\_CollectDestruct](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect)  
*Free up any members associated with an element collection.*
- void [gslc\\_ElemDestruct](#) ([gslc\\_tsElem](#) \*pElem)  
*Free up any members associated with an element.*
- void [gslc\\_ResetFont](#) ([gslc\\_tsFont](#) \*pFont)  
*Initialize a Font struct.*
- void [gslc\\_ResetElem](#) ([gslc\\_tsElem](#) \*pElem)  
*Initialize an Element struct.*

## Variables

- [GSLC\\_CB\\_DEBUG\\_OUT](#) [g\\_pfDebugOut](#)  
*Global debug output function.*

## 9.45.1 Macro Definition Documentation

### 9.45.1.1 GSLC\_2PI

```
#define GSLC_2PI
```

### 9.45.1.2 GSLC\_ALIGN\_BOT\_LEFT

```
#define GSLC_ALIGN_BOT_LEFT
```

Align to bottom-left.

### 9.45.1.3 GSLC\_ALIGN\_BOT\_MID

```
#define GSLC_ALIGN_BOT_MID
```

Align to middle of bottom.

#### 9.45.1.4 GSLC\_ALIGN\_BOT\_RIGHT

```
#define GSLC_ALIGN_BOT_RIGHT
```

Align to bottom-right.

#### 9.45.1.5 GSLC\_ALIGN\_MID\_LEFT

```
#define GSLC_ALIGN_MID_LEFT
```

Align to middle of left side.

#### 9.45.1.6 GSLC\_ALIGN\_MID\_MID

```
#define GSLC_ALIGN_MID_MID
```

Align to center.

#### 9.45.1.7 GSLC\_ALIGN\_MID\_RIGHT

```
#define GSLC_ALIGN_MID_RIGHT
```

Align to middle of right side.

#### 9.45.1.8 GSLC\_ALIGN\_TOP\_LEFT

```
#define GSLC_ALIGN_TOP_LEFT
```

Align to top-left.

#### 9.45.1.9 GSLC\_ALIGN\_TOP\_MID

```
#define GSLC_ALIGN_TOP_MID
```

Align to middle of top.

#### 9.45.1.10 GSLC\_ALIGN\_TOP\_RIGHT

```
#define GSLC_ALIGN_TOP_RIGHT
```

Align to top-right.

#### 9.45.1.11 GSLC\_ALIGNH\_LEFT

```
#define GSLC_ALIGNH_LEFT
```

Horizontal align to left.

#### 9.45.1.12 GSLC\_ALIGNH\_MID

```
#define GSLC_ALIGNH_MID
```

Horizontal align to middle.

#### 9.45.1.13 GSLC\_ALIGNH\_RIGHT

```
#define GSLC_ALIGNH_RIGHT
```

Horizontal align to right.

#### 9.45.1.14 GSLC\_ALIGNV\_BOT

```
#define GSLC_ALIGNV_BOT
```

Vertical align to bottom.

#### 9.45.1.15 GSLC\_ALIGNV\_MID

```
#define GSLC_ALIGNV_MID
```

Vertical align to middle.

**9.45.1.16 GSLC\_ALIGNV\_TOP**

```
#define GSLC_ALIGNV_TOP
```

Element text alignment.

Vertical align to top

**9.45.1.17 GSLC\_COL\_BLACK**

```
#define GSLC_COL_BLACK
```

Black.

**9.45.1.18 GSLC\_COL\_BLUE**

```
#define GSLC_COL_BLUE
```

Blue.

**9.45.1.19 GSLC\_COL\_BLUE\_DK1**

```
#define GSLC_COL_BLUE_DK1
```

Blue (dark1)

**9.45.1.20 GSLC\_COL\_BLUE\_DK2**

```
#define GSLC_COL_BLUE_DK2
```

Blue (dark2)

**9.45.1.21 GSLC\_COL\_BLUE\_DK3**

```
#define GSLC_COL_BLUE_DK3
```

Blue (dark3)

#### 9.45.1.22 GSLC\_COL\_BLUE\_DK4

```
#define GSLC_COL_BLUE_DK4
```

Blue (dark4)

#### 9.45.1.23 GSLC\_COL\_BLUE\_LT1

```
#define GSLC_COL_BLUE_LT1
```

Blue (light1)

#### 9.45.1.24 GSLC\_COL\_BLUE\_LT2

```
#define GSLC_COL_BLUE_LT2
```

Blue (light2)

#### 9.45.1.25 GSLC\_COL\_BLUE\_LT3

```
#define GSLC_COL_BLUE_LT3
```

Blue (light3)

#### 9.45.1.26 GSLC\_COL\_BLUE\_LT4

```
#define GSLC_COL_BLUE_LT4
```

Blue (light4)

#### 9.45.1.27 GSLC\_COL\_BROWN

```
#define GSLC_COL_BROWN
```

Brown.

**9.45.1.28 GSLC\_COL\_CYAN**

```
#define GSLC_COL_CYAN
```

Cyan.

**9.45.1.29 GSLC\_COL\_GRAY**

```
#define GSLC_COL_GRAY
```

Gray.

**9.45.1.30 GSLC\_COL\_GRAY\_DK1**

```
#define GSLC_COL_GRAY_DK1
```

Gray (dark1)

**9.45.1.31 GSLC\_COL\_GRAY\_DK2**

```
#define GSLC_COL_GRAY_DK2
```

Gray (dark2)

**9.45.1.32 GSLC\_COL\_GRAY\_DK3**

```
#define GSLC_COL_GRAY_DK3
```

Gray (dark3)

**9.45.1.33 GSLC\_COL\_GRAY\_DK4**

```
#define GSLC_COL_GRAY_DK4
```

Gray (dark4)

**9.45.1.34 GSLC\_COL\_GRAY\_LT1**

```
#define GSLC_COL_GRAY_LT1
```

Gray (light1)

**9.45.1.35 GSLC\_COL\_GRAY\_LT2**

```
#define GSLC_COL_GRAY_LT2
```

Gray (light2)

**9.45.1.36 GSLC\_COL\_GRAY\_LT3**

```
#define GSLC_COL_GRAY_LT3
```

Gray (light3)

**9.45.1.37 GSLC\_COL\_GRAY\_LT4**

```
#define GSLC_COL_GRAY_LT4
```

Gray (light4)

**9.45.1.38 GSLC\_COL\_GREEN**

```
#define GSLC_COL_GREEN
```

Green.

**9.45.1.39 GSLC\_COL\_GREEN\_DK1**

```
#define GSLC_COL_GREEN_DK1
```

Green (dark1)

**9.45.1.40 GSLC\_COL\_GREEN\_DK2**

```
#define GSLC_COL_GREEN_DK2
```

Green (dark2)

**9.45.1.41 GSLC\_COL\_GREEN\_DK3**

```
#define GSLC_COL_GREEN_DK3
```

Green (dark3)

**9.45.1.42 GSLC\_COL\_GREEN\_DK4**

```
#define GSLC_COL_GREEN_DK4
```

Green (dark4)

**9.45.1.43 GSLC\_COL\_GREEN\_LT1**

```
#define GSLC_COL_GREEN_LT1
```

Green (light1)

**9.45.1.44 GSLC\_COL\_GREEN\_LT2**

```
#define GSLC_COL_GREEN_LT2
```

Green (light2)

**9.45.1.45 GSLC\_COL\_GREEN\_LT3**

```
#define GSLC_COL_GREEN_LT3
```

Green (light3)



**9.45.1.46 GSLC\_COL\_GREEN\_LT4**

```
#define GSLC_COL_GREEN_LT4
```

Green (light4)

**9.45.1.47 GSLC\_COL\_MAGENTA**

```
#define GSLC_COL_MAGENTA
```

Magenta.

**9.45.1.48 GSLC\_COL\_ORANGE**

```
#define GSLC_COL_ORANGE
```

Orange.

**9.45.1.49 GSLC\_COL\_PURPLE**

```
#define GSLC_COL_PURPLE
```

Purple.

**9.45.1.50 GSLC\_COL\_RED**

```
#define GSLC_COL_RED
```

Red.

**9.45.1.51 GSLC\_COL\_RED\_DK1**

```
#define GSLC_COL_RED_DK1
```

Red (dark1)

**9.45.1.52 GSLC\_COL\_RED\_DK2**

```
#define GSLC_COL_RED_DK2
```

Red (dark2)

**9.45.1.53 GSLC\_COL\_RED\_DK3**

```
#define GSLC_COL_RED_DK3
```

Red (dark3)

**9.45.1.54 GSLC\_COL\_RED\_DK4**

```
#define GSLC_COL_RED_DK4
```

Basic color definition.

Red (dark4)

**9.45.1.55 GSLC\_COL\_RED\_LT1**

```
#define GSLC_COL_RED_LT1
```

Red (light1)

**9.45.1.56 GSLC\_COL\_RED\_LT2**

```
#define GSLC_COL_RED_LT2
```

Red (light2)

**9.45.1.57 GSLC\_COL\_RED\_LT3**

```
#define GSLC_COL_RED_LT3
```

Red (light3)

**9.45.1.58 GSLC\_COL\_RED\_LT4**

```
#define GSLC_COL_RED_LT4
```

Red (light4)

**9.45.1.59 GSLC\_COL\_TEAL**

```
#define GSLC_COL_TEAL
```

Teal.

**9.45.1.60 GSLC\_COL\_WHITE**

```
#define GSLC_COL_WHITE
```

White.

**9.45.1.61 GSLC\_COL\_YELLOW**

```
#define GSLC_COL_YELLOW
```

Yellow.

**9.45.1.62 GSLC\_COL\_YELLOW\_DK**

```
#define GSLC_COL_YELLOW_DK
```

Yellow (dark)

**9.45.1.63 GSLC\_COLMONO\_BLACK**

```
#define GSLC_COLMONO_BLACK
```

Black.

**9.45.1.64 GSLC\_COLMONO\_WHITE**

```
#define GSLC_COLMONO_WHITE
```

White.

**9.45.1.65 GSLC\_ELEM\_FEA\_CLICK\_EN**

```
#define GSLC_ELEM_FEA_CLICK_EN
```

Element accepts touch presses.

**9.45.1.66 GSLC\_ELEM\_FEA\_EDIT\_EN**

```
#define GSLC_ELEM_FEA_EDIT_EN
```

Element supports edit.

**9.45.1.67 GSLC\_ELEM\_FEA\_FILL\_EN**

```
#define GSLC_ELEM_FEA_FILL_EN
```

Element is drawn with a fill.

**9.45.1.68 GSLC\_ELEM\_FEA\_FOCUS\_EN**

```
#define GSLC_ELEM_FEA_FOCUS_EN
```

Element can accept focus.

**9.45.1.69 GSLC\_ELEM\_FEA\_FRAME\_EN**

```
#define GSLC_ELEM_FEA_FRAME_EN
```

Element is drawn with a frame.

**9.45.1.70 GSLC\_ELEM\_FEA\_GLOW\_EN**

```
#define GSLC_ELEM_FEA_GLOW_EN
```

Element supports glowing state.

**9.45.1.71 GSLC\_ELEM\_FEA\_NONE**

```
#define GSLC_ELEM_FEA_NONE
```

Element default (no features set))

**9.45.1.72 GSLC\_ELEM\_FEA\_NOSHRINK**

```
#define GSLC_ELEM_FEA_NOSHRINK
```

Element features type.

Element can't be shrunk (eg. contains image)

**9.45.1.73 GSLC\_ELEM\_FEA\_ROUND\_EN**

```
#define GSLC_ELEM_FEA_ROUND_EN
```

Element is drawn with a rounded profile.

**9.45.1.74 GSLC\_ELEM\_FEA\_VALID**

```
#define GSLC_ELEM_FEA_VALID
```

Element record is valid.

**9.45.1.75 GSLC\_ELEMREF\_DEFAULT**

```
#define GSLC_ELEMREF_DEFAULT
```

Define the default element reference flags for new elements.

#### 9.45.1.76 GSLC\_FEATURE\_FOCUS\_ON\_TOUCH

```
#define GSLC_FEATURE_FOCUS_ON_TOUCH
```

#### 9.45.1.77 GSLC\_MAX

```
#define GSLC_MAX(  
    a,  
    b )
```

#### 9.45.1.78 GSLC\_MIN

```
#define GSLC_MIN(  
    a,  
    b )
```

#### 9.45.1.79 GSLC\_PMEM

```
#define GSLC_PMEM
```

### 9.45.2 Typedef Documentation

#### 9.45.2.1 GSLC\_CB\_DEBUG\_OUT

```
typedef int16_t(* GSLC_CB_DEBUG_OUT) (char ch)
```

#### 9.45.2.2 GSLC\_CB\_DRAW

```
typedef bool(* GSLC_CB_DRAW) (void *pvGui, void *pvElemRef, gslc\_teRedrawType eRedraw)
```

Callback function for element drawing.

#### 9.45.2.3 GSLC\_CB\_EVENT

```
typedef bool(* GSLC_CB_EVENT) (void *pvGui, gslc\_tsEvent sEvent)
```

Callback function for element drawing.

#### 9.45.2.4 GSLC\_CB\_INPUT

```
typedef bool(* GSLC_CB_INPUT) (void *pvGui, void *pvElemRef, int16_t nStatus, void *pvData)
```

Callback function for element input ready.

#### 9.45.2.5 GSLC\_CB\_PIN\_POLL

```
typedef bool(* GSLC_CB_PIN_POLL) (void *pvGui, int16_t *pnPinInd, int16_t *pnPinVal)
```

Callback function for pin polling.

#### 9.45.2.6 GSLC\_CB\_TICK

```
typedef bool(* GSLC_CB_TICK) (void *pvGui, void *pvElemRef)
```

Callback function for element tick.

#### 9.45.2.7 GSLC\_CB\_TOUCH

```
typedef bool(* GSLC_CB_TOUCH) (void *pvGui, void *pvElemRef, gslc\_teTouch eTouch, int16_t nX,  
int16_t nY)
```

Callback function for element touch tracking.

#### 9.45.2.8 [gslc\\_tsColor](#)

```
typedef struct gslc\_tsColor gslc\_tsColor
```

Color structure. Defines RGB triplet.

#### 9.45.2.9 gslc\_tsElem

```
typedef struct gslc_tsElem gslc_tsElem
```

Element Struct.

- Represents a single graphic element in the GUIslice environment
- A page is made up of a number of elements
- Each element is created with a user-specified ID for further accesses (or GSLC\_ID\_AUTO for it to be auto-generated)
- Display order of elements in a page is based upon the creation order
- Extensions to the core element types is provided through the pXData reference and pfuncX\* callback functions.

#### 9.45.2.10 gslc\_tsEvent

```
typedef struct gslc_tsEvent gslc_tsEvent
```

Event structure.

#### 9.45.2.11 gslc\_tsEventTouch

```
typedef struct gslc_tsEventTouch gslc_tsEventTouch
```

Structure used to pass touch data through event.

#### 9.45.2.12 gslc\_tsPt

```
typedef struct gslc_tsPt gslc_tsPt
```

Define point coordinates.

#### 9.45.2.13 gslc\_tsRect

```
typedef struct gslc_tsRect gslc_tsRect
```

Rectangular region. Defines X,Y corner coordinates plus dimensions.



## 9.45.2.14 gslc\_tsRectState

```
typedef struct gslc_tsRectState gslc_tsRectState
```

State associated with an element's region.

- This struct is used for [gslc\\_ElemCalcRectState\(\)](#)
- Accounts for various rects including focus, frame and internal content
- Also contains the various colors associated with each region.

## 9.45.3 Enumeration Type Documentation

## 9.45.3.1 gslc\_teAction

```
enum gslc_teAction
```

GUI Action Requested These actions are usually the result of an InputMap lookup.

## Enumerator

GSLC_ACTION_UNDEF	Invalid action.
GSLC_ACTION_NONE	No action to perform.
GSLC_ACTION_FOCUS_PREV	Advance focus to the previous GUI element.
GSLC_ACTION_FOCUS_NEXT	Advance focus to the next GUI element.
GSLC_ACTION_PRESELECT	Pre-Select the currently focused GUI element (glow)
GSLC_ACTION_SELECT	Select the currently focused GUI element.
GSLC_ACTION_SET_REL	Adjust value (relative) of focused element.
GSLC_ACTION_SET_ABS	Adjust value (absolute) of focused element.
GSLC_ACTION_DEBUG	Internal debug action.

## 9.45.3.2 gslc\_teElemId

```
enum gslc_teElemId
```

Element ID enumerations.

- The Element ID is the primary means for user code to reference a graphic element.
- Application code can assign arbitrary Element ID values in the range of 0...16383
- Specifying GSLC\_ID\_AUTO to ElemCreate() requests that GUISlice auto-assign an ID value for the Element. These auto-assigned values will begin at GSLC\_ID\_AUTO\_BASE.
- Negative Element ID values are reserved

**Enumerator**

GSLC_ID_USER_BASE	Starting Element ID for user assignments.
GSLC_ID_NONE	No Element ID has been assigned.
GSLC_ID_AUTO	Auto-assigned Element ID requested.
GSLC_ID_TEMP	ID for Temporary Element.
GSLC_ID_AUTO_BASE	Starting Element ID to start auto-assignment (when GSLC_ID_AUTO is specified)

**9.45.3.3 gslc\_teElemInd**

```
enum gslc_teElemInd
```

Element Index enumerations.

- The Element Index is used for internal purposes as an offset

**Enumerator**

GSLC_IND_NONE	No Element Index is available.
GSLC_IND_FIRST	User elements start at index 0.

**9.45.3.4 gslc\_teElemRefFlags**

```
enum gslc_teElemRefFlags
```

Element reference flags: Describes characteristics of an element.

- Primarily used to support relocation of elements to Flash memory (PROGMEM)

**Enumerator**

GSLC_ELEMREF_NONE	No element defined.
GSLC_ELEMREF_SRC_RAM	Element is read/write Stored in RAM (internal element array)) Access directly.
GSLC_ELEMREF_SRC_PROG	Element is read-only / const Stored in FLASH (external to element array) Access via PROGMEM.
GSLC_ELEMREF_SRC_CONST	Element is read-only / const Stored in FLASH (external to element array) Access directly.
GSLC_ELEMREF_REDRAW_NONE	No redraw requested.
GSLC_ELEMREF_REDRAW_FULL	Full redraw of element requested.
GSLC_ELEMREF_REDRAW_INC	Incremental redraw of element requested.
GSLC_ELEMREF_REDRAW_FOCUS	Focus-only redraw of element requested.

## Enumerator

GSLC_ELEMREF_EDITING	Element is in edit state (1=edit, 0=navigate)
GSLC_ELEMREF_FOCUSED	Element state is focused.
GSLC_ELEMREF_GLOWING	Element state is glowing.
GSLC_ELEMREF_VISIBLE	Element is currently shown (ie. visible)
GSLC_ELEMREF_SRC	Mask for Source flags [bits 1,0].
GSLC_ELEMREF_REDRAW_MASK	Mask for Redraw flags [bits 5,4].

## 9.45.3.5 gslc\_teEventSubType

```
enum gslc\_teEventSubType
```

Event sub-types.

## Enumerator

GSLC_EVTSUB_NONE	
GSLC_EVTSUB_DRAW_NEEDED	Incremental redraw (as needed)
GSLC_EVTSUB_DRAW_FORCE	Force a full redraw.

## 9.45.3.6 gslc\_teEventType

```
enum gslc\_teEventType
```

Event types.

## Enumerator

GSLC_EVT_NONE	No event; ignore.
GSLC_EVT_DRAW	Perform redraw.
GSLC_EVT_TOUCH	Track touch event.
GSLC_EVT_TICK	Perform background tick handling.
GSLV_EVT_CUSTOM	Custom event.

## 9.45.3.7 gslc\_teFontId

```
enum gslc\_teFontId
```

Font ID enumerations.

- The Font ID is the primary means for user code to reference a specific font.
- Application code can assign arbitrary Font ID values in the range of 0...16383
- Negative Font ID values are reserved

#### Enumerator

GSLC_FONT_USER_BASE	Starting Font ID for user assignments.
GSLC_FONT_NONE	No Font ID has been assigned.

#### 9.45.3.8 gslc\_teFontRefMode

enum [gslc\\_teFontRefMode](#)

Font Reference modes.

- The Font Reference mode defines the source for the selected font. For graphics libraries that offer multiple types of fonts, this can be used to differentiate between a default font, hardware fonts, software fonts, etc.
- The encoding between the different modes is driver-specific.

#### Enumerator

GSLC_FONTREF_MODE_DEFAULT	Default font mode.
GSLC_FONTREF_MODE_1	Font mode 1.
GSLC_FONTREF_MODE_2	Font mode 2.
GSLC_FONTREF_MODE_3	Font mode 3.

#### 9.45.3.9 gslc\_teFontRefType

enum [gslc\\_teFontRefType](#)

Font Reference types.

- The Font Reference type defines the way in which a font is selected. In some device targets (such as LINUX SDL) a filename to a font file is provided. In others (such as Arduino, ESP8266), a pointer is given to a font structure (or NULL for default).

#### Enumerator

GSLC_FONTREF_FNAME	Font reference is a filename (full path)
GSLC_FONTREF_PTR	Font reference is a pointer to a font structure.

## 9.45.3.10 gslc\_teGroupId

enum `gslc_teGroupId`

Group ID enumerations.

## Enumerator

GSLC_GROUP_ID_USER_BASE	Starting Group ID for user assignments.
GSLC_GROUP_ID_NONE	No Group ID has been assigned.

## 9.45.3.11 gslc\_telmgRefFlags

enum `gslc_telmgRefFlags`

Image reference flags: Describes characteristics of an image reference.

## Enumerator

GSLC_IMGREF_NONE	No image defined.
GSLC_IMGREF_SRC_FILE	Image is stored in file system.
GSLC_IMGREF_SRC_SD	Image is stored on SD card.
GSLC_IMGREF_SRC_RAM	Image is stored in RAM.
GSLC_IMGREF_SRC_PROG	Image is stored in program memory (PROGMEM)
GSLC_IMGREF_FMT_BMP24	Image format is BMP (24-bit)
GSLC_IMGREF_FMT_BMP16	Image format is BMP (16-bit RGB565)
GSLC_IMGREF_FMT_RAW1	Image format is raw monochrome (1-bit)
GSLC_IMGREF_FMT_JPG	Image format is JPG (ESP32/ESP8366)
GSLC_IMGREF_SRC	Mask for Source flags.
GSLC_IMGREF_FMT	Mask for Format flags.

## 9.45.3.12 gslc\_telnitStat

enum `gslc_telnitStat`

Status of a module's initialization.

## Enumerator

GSLC_INITSTAT_UNDEF	Module status has not been defined yet.
GSLC_INITSTAT_INACTIVE	Module is not enabled.
GSLC_INITSTAT_FAIL	Module is enabled but failed to init.
GSLC_INITSTAT_ACTIVE	Module is enabled and initialized OK.

### 9.45.3.13 gslc\_teInputMode

enum `gslc_teInputMode`

External input mode.

Dictates how directional controls affect the interaction with the GUI elements.

#### Enumerator

GSLC_INPUTMODE_NAV	External input is in navigation mode.
GSLC_INPUTMODE_EDIT	External input is in element edit mode.

### 9.45.3.14 gslc\_teInputRawEvent

enum `gslc_teInputRawEvent`

Raw input event types: touch, key, GPIOs.

#### Enumerator

GSLC_INPUT_NONE	No input event.
GSLC_INPUT_TOUCH	Touch / mouse event.
GSLC_INPUT_KEY_DOWN	Key press down / pin input asserted.
GSLC_INPUT_KEY_UP	Key press up (released)
GSLC_INPUT_PIN_ASSERT	GPIO pin input asserted (eg. set to 1 / High)
GSLC_INPUT_PIN_DEASSERT	GPIO pin input deasserted (eg. set to 0 / Low)

### 9.45.3.15 gslc\_tePageId

enum `gslc_tePageId`

Page ID enumerations.

- The Page ID is the primary means for user code to reference a specific page of elements.
- Application code can assign arbitrary Page ID values in the range of 0...16383
- Negative Page ID values are reserved

## Enumerator

GSLC_PAGE_USER_BASE	Starting Page ID for user assignments.
GSLC_PAGE_NONE	No Page ID has been assigned.

## 9.45.3.16 gslc\_tePin

```
enum gslc_tePin
```

General purpose pin/button constants.

## Enumerator

GSLC_PIN_BTN_A	Button A (short press)
GSLC_PIN_BTN_A_LONG	Button A (long press)
GSLC_PIN_BTN_B	Button B (short press)
GSLC_PIN_BTN_B_LONG	Button B (long press)
GSLC_PIN_BTN_C	Button C (short press)
GSLC_PIN_BTN_C_LONG	Button C (long press)
GSLC_PIN_BTN_D	Button D (short press)
GSLC_PIN_BTN_D_LONG	Button D (long press)
GSLC_PIN_BTN_E	Button E (short press)
GSLC_PIN_BTN_E_LONG	Button E (long press)
GSLC_PIN_BTN_UP	Button Up (short press)
GSLC_PIN_BTN_DOWN	Button Down (short press)
GSLC_PIN_BTN_LEFT	Button Left (short press)
GSLC_PIN_BTN_RIGHT	Button Right (short press)
GSLC_PIN_BTN_SEL	Button Select (short press)

## 9.45.3.17 gslc\_teRedrawType

```
enum gslc_teRedrawType
```

Redraw types.

## Enumerator

GSLC_REDRAW_NONE	No redraw requested.
GSLC_REDRAW_FULL	Full redraw of element requested.
GSLC_REDRAW_INC	Incremental redraw of element requested.
GSLC_REDRAW_FOCUS	Only focus redraw requested.

#### 9.45.3.18 gslc\_teStackPage

enum `gslc_teStackPage`

Define page stack.

##### Enumerator

GSLC_STACK_BASE	Base page.
GSLC_STACK_CUR	Current page.
GSLC_STACK_OVERLAY	Overlay page (eg. popups)
GSLC_STACK_MAX	Defines maximum number of pages in stack.

#### 9.45.3.19 gslc\_teTouch

enum `gslc_teTouch`

Processed event from input raw events and actions.

##### Enumerator

GSLC_TOUCH_NONE	No touch event active.
GSLC_TOUCH_TYPE_MASK	Mask for type: coord/direct mode.
GSLC_TOUCH_COORD	Event based on touch coordinate.
GSLC_TOUCH_DIRECT	Event based on specific element index (keyboard/GPIO action)
GSLC_TOUCH_SUBTYPE_MASK	Mask for subtype.
GSLC_TOUCH_DOWN	Touch event (down)
GSLC_TOUCH_DOWN_IN	Touch event (down inside tracked element)
GSLC_TOUCH_DOWN_OUT	Touch event (down outside tracked element)
GSLC_TOUCH_UP	Touch event (up)
GSLC_TOUCH_UP_IN	Touch event (up inside tracked element)
GSLC_TOUCH_UP_OUT	Touch event (up outside tracked element)
GSLC_TOUCH_MOVE	Touch event (move)
GSLC_TOUCH_MOVE_IN	Touch event (move inside tracked element)
GSLC_TOUCH_MOVE_OUT	Touch event (move outside tracked element)
GSLC_TOUCH_FOCUS_ON	Direct event focus on element.
GSLC_TOUCH_FOCUS_OFF	Direct event focus away from focused element.
GSLC_TOUCH_FOCUS_PRESELECT	Direct event select focus element (glow before select)
GSLC_TOUCH_FOCUS_SELECT	Direct event select focus element.
GSLC_TOUCH_SET_REL	Direct event set value (relative) on focus element.
GSLC_TOUCH_SET_ABS	Direct event set value (absolute) on focus element.

#### 9.45.3.20 gslc\_teTxtFlags

enum `gslc_teTxtFlags`



Text reference flags: Describes the characteristics of a text string (ie.

whether internal to element or external and RAM vs Flash).)

Supported flag combinations are:

- ALLOC\_NONE
- ALLOC\_INT | MEM\_RAM
- ALLOC\_EXT | MEM\_RAM
- ALLOC\_EXT | MEM\_PROG

#### Enumerator

GSLC_TXT_MEM_RAM	Text string is in SRAM (read-write)
GSLC_TXT_MEM_PROG	Text string is in PROGMEM (read-only)
GSLC_TXT_ALLOC_NONE	No text string present.
GSLC_TXT_ALLOC_INT	Text string allocated in internal element memory (GSLC_STR_LOCAL=1)
GSLC_TXT_ALLOC_EXT	Text string allocated in external memory (GSLC_STR_LOCAL=0), ie. user code.
GSLC_TXT_ENC_PLAIN	Encoding is plain text (LATIN1))
GSLC_TXT_ENC_UTF8	Encoding is UTF-8.
GSLC_TXT_MEM	Mask for updating text memory type.
GSLC_TXT_ALLOC	Mask for updating location of text string buffer allocation.
GSLC_TXT_ENC	Mask for updating text encoding.
GSLC_TXT_DEFAULT	

#### 9.45.3.21 gslc\_teTypeCore

```
enum gslc\_teTypeCore
```

Element type.

#### Enumerator

GSLC_TYPE_NONE	No element type specified.
GSLC_TYPE_BKGND	Background element type.
GSLC_TYPE_BTN	Button element type.
GSLC_TYPE_TXT	Text label element type.
GSLC_TYPE_BOX	Box / frame element type.
GSLC_TYPE_LINE	Line element type.
GSLC_TYPE_BASE_EXTEND	Base value for extended type enumerations.

### 9.45.4 Variable Documentation

#### 9.45.4.1 g\_pfDebugOut

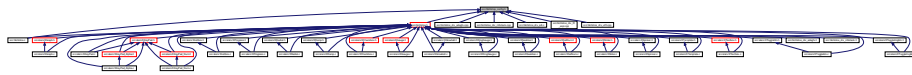
`GSLC_CB_DEBUG_OUT` `g_pfDebugOut`

Global debug output function.

- The user assigns this function via `gslc_InitDebug()`

### 9.46 src/GUIslice\_config.h File Reference

This graph shows which files directly or indirectly include this file:



### 9.47 src/GUIslice\_drv.h File Reference

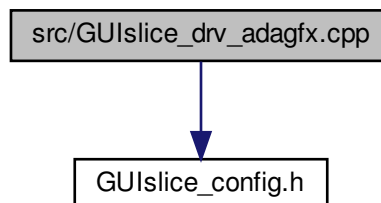
This graph shows which files directly or indirectly include this file:



### 9.48 src/GUIslice\_drv\_adagfx.cpp File Reference

```
#include "GUIslice_config.h"
```

Include dependency graph for `GUISlice_drv_adagfx.cpp`:



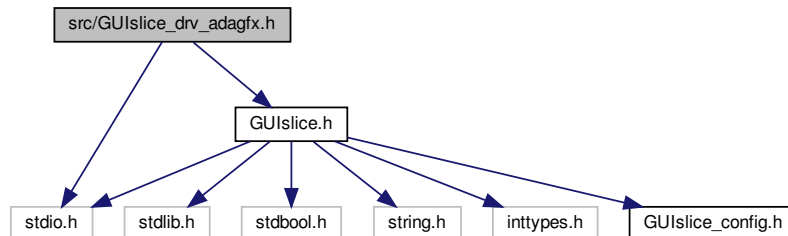
## 9.49 src/GUISlice\_drv\_adagfx.h File Reference

GUISlice library (driver layer for Adafruit-GFX)

```
#include "GUISlice.h"
```

```
#include <stdio.h>
```

Include dependency graph for GUISlice\_drv\_adagfx.h:



### Data Structures

- struct [gslc\\_tsDriver](#)

### Macros

- #define [DRV\\_HAS\\_DRAW\\_POINT](#)  
Support [gslc\\_DrvDrawPoint\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_POINTS](#)  
Support [gslc\\_DrvDrawPoints\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_LINE](#)  
Support [gslc\\_DrvDrawLine\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_RECT\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameRect\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_RECT\\_FILL](#)  
Support [gslc\\_DrvDrawFillRect\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_RECT\\_ROUND\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_RECT\\_ROUND\\_FILL](#)  
Support [gslc\\_DrvDrawFillRoundRect\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_CIRCLE\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameCircle\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_CIRCLE\\_FILL](#)  
Support [gslc\\_DrvDrawFillCircle\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_TRI\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameTriangle\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_TRI\\_FILL](#)  
Support [gslc\\_DrvDrawFillTriangle\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_TEXT](#)  
Support [gslc\\_DrvDrawTxt\(\)](#)
- #define [DRV\\_HAS\\_DRAW\\_BMP\\_MEM](#)  
Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)
- #define [DRV\\_OVERRIDE\\_TXT\\_ALIGN](#)  
Driver provides text alignment.

## Functions

- bool [gslc\\_DrvInit](#) ([gslc\\_tsGui](#) \*pGui)  
*Initialize the SDL library.*
- bool [gslc\\_DrvInitTs](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)  
*Perform any touchscreen-specific initialization.*
- void [gslc\\_DrvDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Free up any members associated with the driver.*
- const char \* [gslc\\_DrvGetNameDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the display driver name.*
- const char \* [gslc\\_DrvGetNameTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the touch driver name.*
- void \* [gslc\\_DrvGetDriverDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native display driver instance.*
- void \* [gslc\\_DrvGetDriverTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native touch driver instance.*
- void \* [gslc\\_DrvLoadImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsImgRef](#) sImgRef)  
*Load a bitmap (\*.bmp) and create a new image resource.*
- bool [gslc\\_DrvSetBkgndImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsImgRef](#) sImgRef)  
*Configure the background to use a bitmap image.*
- bool [gslc\\_DrvSetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the background to use a solid color.*
- bool [gslc\\_DrvSetElemImageNorm](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)  
*Set an element's normal-state image.*
- bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)  
*Set an element's glow-state image.*
- void [gslc\\_DrvImageDestruct](#) (void \*pvImg)  
*Release an image surface.*
- bool [gslc\\_DrvSetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for future drawing updates.*
- const void \* [gslc\\_DrvFontAdd](#) ([gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)  
*Load a font from a resource and return pointer to it.*
- void [gslc\\_DrvFontsDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Release all fonts defined in the GUI.*
- bool [gslc\\_DrvGetTxtSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, int16\_t \*pnTxtX, int16\_t \*pnTxtY, uint16\_t \*pnTxtSzW, uint16\_t \*pnTxtSzH)  
*Get the extent (width and height) of a text string.*
- bool [gslc\\_DrvDrawTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nTxtX, int16\_t nTxtY, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)  
*Draw a text string at the given coordinate.*
- void [gslc\\_DrvPageFlipNow](#) ([gslc\\_tsGui](#) \*pGui)  
*Force a page flip to occur.*
- bool [gslc\\_DrvDrawPoint](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawPoints](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*asPt, uint16\_t nNumPt, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a framed rectangle.*
- bool [gslc\\_DrvDrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a filled rectangle.*

- bool [gslc\\_DrvDrawFrameRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
  - Draw a framed rounded rectangle.*
- bool [gslc\\_DrvDrawFillRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
  - Draw a filled rounded rectangle.*
- bool [gslc\\_DrvDrawLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, [gslc\\_tsColor](#) nCol)
  - Draw a line.*
- bool [gslc\\_DrvDrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
  - Draw a framed circle.*
- bool [gslc\\_DrvDrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
  - Draw a filled circle.*
- bool [gslc\\_DrvDrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
  - Draw a framed triangle.*
- bool [gslc\\_DrvDrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
  - Draw a filled triangle.*
- bool [gslc\\_DrvDrawImage](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, [gslc\\_tsImgRef](#) slmgRef)
  - Copy all of source image to destination screen at specified coordinate.*
- void [gslc\\_DrvDrawMonoFromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)
  - Draw a monochrome bitmap from a memory array.*
- void [gslc\\_DrvDrawBmp24FromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)
  - Draw a color 24-bit depth bitmap from a memory array.*
- void [gslc\\_DrvDrawBmp24FromSD](#) ([gslc\\_tsGui](#) \*pGui, const char \*filename, uint16\_t x, uint16\_t y)
  - Draw a color 24-bit depth bitmap from SD card.*
- void [gslc\\_DrvDrawBkgnd](#) ([gslc\\_tsGui](#) \*pGui)
  - Copy the background image to destination screen.*
- bool [gslc\\_DrvInitTouch](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)
  - Perform any touchscreen-specific initialization.*
- bool [gslc\\_DrvGetTouch](#) ([gslc\\_tsGui](#) \*pGui, int16\_t \*pnX, int16\_t \*pnY, uint16\_t \*pnPress, [gslc\\_teInputRawEvent](#) \*peInputEvent, int16\_t \*pnInputVal)
  - Get the last touch event from the internal touch handler.*
- bool [gslc\\_DrvRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)
  - Change rotation, automatically adapt touchscreen axes swap/flip.*
- uint16\_t [gslc\\_DrvAdaptColorToRaw](#) ([gslc\\_tsColor](#) nCol)

### 9.49.1 Detailed Description

GUISlice library (driver layer for Adafruit-GFX)

### 9.49.2 Macro Definition Documentation

#### 9.49.2.1 DRV\_HAS\_DRAW\_BMP\_MEM

```
#define DRV_HAS_DRAW_BMP_MEM
```

Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)

#### 9.49.2.2 DRV\_HAS\_DRAW\_CIRCLE\_FILL

```
#define DRV_HAS_DRAW_CIRCLE_FILL
```

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.49.2.3 DRV\_HAS\_DRAW\_CIRCLE\_FRAME

```
#define DRV_HAS_DRAW_CIRCLE_FRAME
```

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.49.2.4 DRV\_HAS\_DRAW\_LINE

```
#define DRV_HAS_DRAW_LINE
```

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.49.2.5 DRV\_HAS\_DRAW\_POINT

```
#define DRV_HAS_DRAW_POINT
```

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.49.2.6 DRV\_HAS\_DRAW\_POINTS

```
#define DRV_HAS_DRAW_POINTS
```

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.49.2.7 DRV\_HAS\_DRAW\_RECT\_FILL

```
#define DRV_HAS_DRAW_RECT_FILL
```

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.49.2.8 DRV\_HAS\_DRAW\_RECT\_FRAME

```
#define DRV_HAS_DRAW_RECT_FRAME
```

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.49.2.9 DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

```
#define DRV_HAS_DRAW_RECT_ROUND_FILL
```

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.49.2.10 DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

```
#define DRV_HAS_DRAW_RECT_ROUND_FRAME
```

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.49.2.11 DRV\_HAS\_DRAW\_TEXT

```
#define DRV_HAS_DRAW_TEXT
```

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.49.2.12 DRV\_HAS\_DRAW\_TRI\_FILL

```
#define DRV_HAS_DRAW_TRI_FILL
```

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

#### 9.49.2.13 DRV\_HAS\_DRAW\_TRI\_FRAME

```
#define DRV_HAS_DRAW_TRI_FRAME
```

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

#### 9.49.2.14 DRV\_OVERRIDE\_TXT\_ALIGN

```
#define DRV_OVERRIDE_TXT_ALIGN
```

Driver provides text alignment.

### 9.49.3 Function Documentation

#### 9.49.3.1 gslc\_DrvAdaptColorToRaw()

```
uint16_t gslc_DrvAdaptColorToRaw (
    gslc\_tsColor nCol )
```

#### 9.49.3.2 gslc\_DrvDestruct()

```
void gslc_DrvDestruct (
    gslc\_tsGui * pGui )
```

Free up any members associated with the driver.

- Eg. renderers, windows, background surfaces, etc.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

none



## 9.49.3.3 gslc\_DrvDrawBkgnd()

```
void gslc_DrvDrawBkgnd (
    gslc_tsGui * pGui )
```

Copy the background image to destination screen.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

true if success, false if fail

## 9.49.3.4 gslc\_DrvDrawBmp24FromMem()

```
void gslc_DrvDrawBmp24FromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a color 24-bit depth bitmap from a memory array.

- Note that users must convert images from their native format (eg. BMP, PNG, etc.) into a C array. Please refer to the following guide for details: <https://github.com/ImpulseAdventure/GUIslice/wiki/Display-Images-from-FLASH>
- The converted file (c array) can then be included in the sketch.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	X coord for copy
in	<i>nDstY</i>	Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

## Returns

none

#### 9.49.3.5 gslc\_DrvDrawBmp24FromSD()

```
void gslc_DrvDrawBmp24FromSD (
    gslc_tsGui * pGui,
    const char * filename,
    uint16_t x,
    uint16_t y )
```

Draw a color 24-bit depth bitmap from SD card.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>filename</i>	Filename on SD card (usually in form "/pic.bmp")
in	<i>x</i>	X coordinate to draw bitmap
in	<i>y</i>	Y coordinate to draw bitmap

##### Returns

none

#### 9.49.3.6 gslc\_DrvDrawFillCircle()

```
bool gslc_DrvDrawFillCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled circle.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to fill

##### Returns

true if success, false if error

## 9.49.3.7 gslc\_DrvDrawFillRect()

```
bool gslc_DrvDrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

## Returns

true if success, false if error

## 9.49.3.8 gslc\_DrvDrawFillRoundRect()

```
bool gslc_DrvDrawFillRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled rounded rectangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to fill

## Returns

true if success, false if error

## 9.49.3.9 gslc\_DrvDrawFillTriangle()

```
bool gslc_DrvDrawFillTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
```

```

    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )

```

Draw a filled triangle.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to fill

#### Returns

true if success, false if error

#### 9.49.3.10 gslc\_DrvDrawFrameCircle()

```

bool gslc_DrvDrawFrameCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )

```

Draw a framed circle.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to frame

#### Returns

true if success, false if error

#### 9.49.3.11 gslc\_DrvDrawFrameRect()

```
bool gslc_DrvDrawFrameRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a framed rectangle.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value to frame

##### Returns

true if success, false if error

#### 9.49.3.12 gslc\_DrvDrawFrameRoundRect()

```
bool gslc_DrvDrawFrameRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed rounded rectangle.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to frame

##### Returns

true if success, false if error

#### 9.49.3.13 gslc\_DrvDrawFrameTriangle()

```
bool gslc_DrvDrawFrameTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
```

```

    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )

```

Draw a framed triangle.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to frame

#### Returns

true if success, false if error

#### 9.49.3.14 gslc\_DrvDrawImage()

```

bool gslc_DrvDrawImage (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    gslc_tsImgRef sImgRef )

```

Copy all of source image to destination screen at specified coordinate.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>sImgRef</i>	Image reference

#### Returns

true if success, false if fail

## 9.49.3.15 gslc\_DrvDrawLine()

```
bool gslc_DrvDrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw a line.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	Line start (X coordinate)
in	<i>nY0</i>	Line start (Y coordinate)
in	<i>nX1</i>	Line finish (X coordinate)
in	<i>nY1</i>	Line finish (Y coordinate)
in	<i>nCol</i>	Color RGB value to draw

## Returns

true if success, false if error

## 9.49.3.16 gslc\_DrvDrawMonoFromMem()

```
void gslc_DrvDrawMonoFromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a monochrome bitmap from a memory array.

- Draw from the bitmap buffer using the foreground color defined in the header (unset bits are transparent)

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

**Returns**

none

**9.49.3.17 gslc\_DrvDrawPoint()**

```
bool gslc_DrvDrawPoint (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

**9.49.3.18 gslc\_DrvDrawPoints()**

```
bool gslc_DrvDrawPoints (
    gslc_tsGui * pGui,
    gslc_tsPt * asPt,
    uint16_t nNumPt,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>asPt</i>	Array of points to draw
in	<i>n</i> $\leftrightarrow$ <i>NumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error



## 9.49.3.19 gslc\_DrvDrawTxt()

```
bool gslc_DrvDrawTxt (
    gslc_tsGui * pGui,
    int16_t nTxtX,
    int16_t nTxtY,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string at the given coordinate.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nTxtX</i>	X coordinate of top-left text string
in	<i>nTxtY</i>	Y coordinate of top-left text string
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	unused in ADAGFX, defaults to black

## Returns

true if success, false if failure

## 9.49.3.20 gslc\_DrvFontAdd()

```
const void* gslc_DrvFontAdd (
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font from a resource and return pointer to it.

## Parameters

in	<i>eFontRefType</i>	Font reference type (GSLC_FONTREF_PTR for Arduino)
in	<i>pvFontRef</i>	Font reference pointer (Pointer to the GFXFont array)
in	<i>nFontSz</i>	Typeface size to use

**Returns**

Void ptr to driver-specific font if load was successful, NULL otherwise

**9.49.3.21 gslc\_DrvFontsDestruct()**

```
void gslc_DrvFontsDestruct (
    gslc_tsGui * pGui )
```

Release all fonts defined in the GUI.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.49.3.22 gslc\_DrvGetDriverDisp()**

```
void* gslc_DrvGetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.49.3.23 gslc\_DrvGetDriverTouch()**

```
void* gslc_DrvGetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.49.3.24 gslc\_DrvGetNameDisp()**

```
const char* gslc_DrvGetNameDisp (  
    gslc_tsGui * pGui )
```

Get the display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.49.3.25 gslc\_DrvGetNameTouch()**

```
const char* gslc_DrvGetNameTouch (  
    gslc_tsGui * pGui )
```

Get the touch driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

### 9.49.3.26 gslc\_DrvGetTouch()

```
bool gslc_DrvGetTouch (
    gslc_tsGui * pGui,
    int16_t * pnX,
    int16_t * pnY,
    uint16_t * pnPress,
    gslc_teInputRawEvent * peInputEvent,
    int16_t * pnInputVal )
```

Get the last touch event from the internal touch handler.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
out	<i>pnX</i>	Ptr to X coordinate of last touch event
out	<i>pnY</i>	Ptr to Y coordinate of last touch event
out	<i>pnPress</i>	Ptr to Pressure level of last touch event (0 for none, 1 for touch)
out	<i>peInputEvent</i>	Indication of event type
out	<i>pnInputVal</i>	Additional data for event type

#### Returns

true if an event was detected or false otherwise

### 9.49.3.27 gslc\_DrvGetTxtSize()

```
bool gslc_DrvGetTxtSize (
    gslc_tsGui * pGui,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    int16_t * pnTxtX,
    int16_t * pnTxtY,
    uint16_t * pnTxtSzW,
    uint16_t * pnTxtSzH )
```

Get the extent (width and height) of a text string.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pFont</i>	Ptr to Font structure
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
out	<i>pnTxtX</i>	Ptr to offset X of text
out	<i>pnTxtY</i>	Ptr to offset Y of text
out	<i>pnTxtSzW</i>	Ptr to width of text
out	<i>pnTxtSzH</i>	Ptr to height of text

**Returns**

true if success, false if failure

**9.49.3.28 gslc\_DrvImageDestruct()**

```
void gslc_DrvImageDestruct (
    void * pVImg )
```

Release an image surface.

**Parameters**

in	<i>pVImg</i>	Void ptr to image
----	--------------	-------------------

**Returns**

none

**9.49.3.29 gslc\_DrvInit()**

```
bool gslc_DrvInit (
    gslc_tsGui * pGui )
```

Initialize the SDL library.

- Performs clean startup workaround (if enabled)
- Configures video mode
- Initializes font support

**PRE:**

- The environment variables should be configured before calling [gslc\\_DrvInit\(\)](#). This can be done with `gslc_↔DrvInitEnv()` or manually in user function.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

true if success, false if fail

#### 9.49.3.30 gslc\_DrvInitTouch()

```
bool gslc_DrvInitTouch (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

##### Returns

true if successful

#### 9.49.3.31 gslc\_DrvInitTs()

```
bool gslc_DrvInitTs (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

##### Returns

true if successful

#### 9.49.3.32 gslc\_DrvLoadImage()

```
void* gslc_DrvLoadImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Load a bitmap (\*.bmp) and create a new image resource.

Transparency is enabled by `GSLC_BMP_TRANS_EN` through use of color (`GSLC_BMP_TRANS_RGB`).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

Image pointer (surface/texture) or NULL if error

**9.49.3.33 gslc\_DrvPageFlipNow()**

```
void gslc_DrvPageFlipNow (  
    gslc_tsGui * pGui )
```

Force a page flip to occur.

This generally copies active screen surface to the display.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.49.3.34 gslc\_DrvRotate()**

```
bool gslc_DrvRotate (  
    gslc_tsGui * pGui,  
    uint8_t nRotation )
```

Change rotation, automatically adapt touchscreen axes swap/flip.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

**Returns**

true if successful

#### 9.49.3.35 gslc\_DrvSetBkgndColor()

```
bool gslc_DrvSetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

##### Returns

true if success, false if fail

#### 9.49.3.36 gslc\_DrvSetBkgndImage()

```
bool gslc_DrvSetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

##### Returns

true if success, false if fail

#### 9.49.3.37 gslc\_DrvSetClipRect()

```
bool gslc_DrvSetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for future drawing updates.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Rectangular region to constrain edits

## Returns

true if success, false if error

## 9.49.3.38 gslc\_DrvSetElemImageGlow()

```
bool gslc_DrvSetElemImageGlow (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's glow-state image.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

## Returns

true if success, false if error

## 9.49.3.39 gslc\_DrvSetElemImageNorm()

```
bool gslc_DrvSetElemImageNorm (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's normal-state image.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

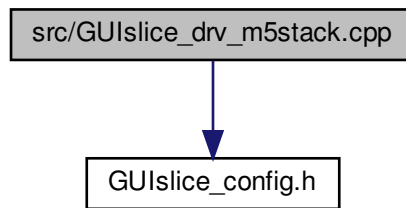
**Returns**

true if success, false if error

## 9.50 src/GUISlice\_drv\_m5stack.cpp File Reference

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_m5stack.cpp:



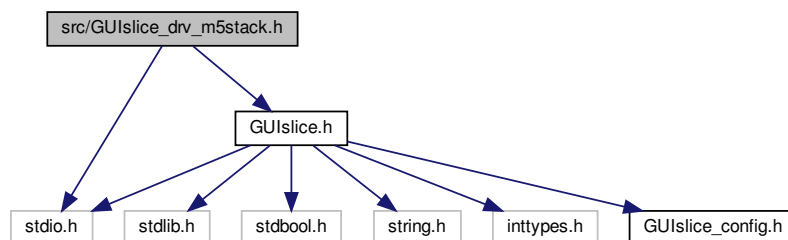
## 9.51 src/GUISlice\_drv\_m5stack.h File Reference

GUISlice library (driver layer for M5stack)

```
#include "GUISlice.h"
```

```
#include <stdio.h>
```

Include dependency graph for GUISlice\_drv\_m5stack.h:



### Data Structures

- struct [gslc\\_tsDriver](#)

## Macros

- `#define DRV_HAS_DRAW_POINT`  
Support `gslc_DrvDrawPoint()`
- `#define DRV_HAS_DRAW_POINTS`  
Support `gslc_DrvDrawPoints()`
- `#define DRV_HAS_DRAW_LINE`  
Support `gslc_DrvDrawLine()`
- `#define DRV_HAS_DRAW_RECT_FRAME`  
Support `gslc_DrvDrawFrameRect()`
- `#define DRV_HAS_DRAW_RECT_FILL`  
Support `gslc_DrvDrawFillRect()`
- `#define DRV_HAS_DRAW_RECT_ROUND_FRAME`  
Support `gslc_DrvDrawFrameRoundRect()`
- `#define DRV_HAS_DRAW_RECT_ROUND_FILL`  
Support `gslc_DrvDrawFillRoundRect()`
- `#define DRV_HAS_DRAW_CIRCLE_FRAME`  
Support `gslc_DrvDrawFrameCircle()`
- `#define DRV_HAS_DRAW_CIRCLE_FILL`  
Support `gslc_DrvDrawFillCircle()`
- `#define DRV_HAS_DRAW_TRI_FRAME`  
Support `gslc_DrvDrawFrameTriangle()`
- `#define DRV_HAS_DRAW_TRI_FILL`  
Support `gslc_DrvDrawFillTriangle()`
- `#define DRV_HAS_DRAW_TEXT`  
Support `gslc_DrvDrawTxt()`
- `#define DRV_HAS_DRAW_BMP_MEM`  
Support `gslc_DrvDrawBmp24FromMem()`
- `#define DRV_OVERRIDE_TXT_ALIGN`  
Driver provides text alignment.

## Functions

- `bool gslc_DrvInit (gslc_tsGui *pGui)`  
Initialize the SDL library.
- `bool gslc_DrvInitTs (gslc_tsGui *pGui, const char *acDev)`  
Perform any touchscreen-specific initialization.
- `void gslc_DrvDestruct (gslc_tsGui *pGui)`  
Free up any members associated with the driver.
- `const char * gslc_DrvGetNameDisp (gslc_tsGui *pGui)`  
Get the display driver name.
- `const char * gslc_DrvGetNameTouch (gslc_tsGui *pGui)`  
Get the touch driver name.
- `void * gslc_DrvGetDriverDisp (gslc_tsGui *pGui)`  
Get the native display driver instance.
- `void * gslc_DrvGetDriverTouch (gslc_tsGui *pGui)`  
Get the native touch driver instance.
- `void * gslc_DrvLoadImage (gslc_tsGui *pGui, gslc_tsImgRef sImgRef)`  
Load a bitmap (\*.bmp) and create a new image resource.
- `bool gslc_DrvSetBkgndImage (gslc_tsGui *pGui, gslc_tsImgRef sImgRef)`

- Configure the background to use a bitmap image.*

  - bool [gslc\\_DrvSetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)
- Configure the background to use a solid color.*

  - bool [gslc\\_DrvSetElemImageNorm](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)
- Set an element's normal-state image.*

  - bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)
- Set an element's glow-state image.*

  - void [gslc\\_DrvImageDestruct](#) (void \*pvImg)
- Release an image surface.*

  - bool [gslc\\_DrvSetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)
- Set the clipping rectangle for future drawing updates.*

  - const void \* [gslc\\_DrvFontAdd](#) ([gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)
- Load a font from a resource and return pointer to it.*

  - void [gslc\\_DrvFontsDestruct](#) ([gslc\\_tsGui](#) \*pGui)
- Release all fonts defined in the GUI.*

  - bool [gslc\\_DrvGetTxtSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, int16\_t \*pnTxtX, int16\_t \*pnTxtY, uint16\_t \*pnTxtSzW, uint16\_t \*pnTxtSzH)
- Get the extent (width and height) of a text string.*

  - bool [gslc\\_DrvDrawTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nTxtX, int16\_t nTxtY, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)
- Draw a text string at the given coordinate.*

  - bool [gslc\\_DrvDrawTxtAlign](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int8\_t eTxtAlign, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)
- Draw a text string in a bounding box using the specified alignment.*

  - void [gslc\\_DrvPageFlipNow](#) ([gslc\\_tsGui](#) \*pGui)
- Force a page flip to occur.*

  - bool [gslc\\_DrvDrawPoint](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, [gslc\\_tsColor](#) nCol)
- Draw a point.*

  - bool [gslc\\_DrvDrawPoints](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*asPt, uint16\_t nNumPt, [gslc\\_tsColor](#) nCol)
- Draw a point.*

  - bool [gslc\\_DrvDrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)
- Draw a framed rectangle.*

  - bool [gslc\\_DrvDrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)
- Draw a filled rectangle.*

  - bool [gslc\\_DrvDrawFrameRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a framed rounded rectangle.*

  - bool [gslc\\_DrvDrawFillRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a filled rounded rectangle.*

  - bool [gslc\\_DrvDrawLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, [gslc\\_tsColor](#) nCol)
- Draw a line.*

  - bool [gslc\\_DrvDrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a framed circle.*

  - bool [gslc\\_DrvDrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a filled circle.*

  - bool [gslc\\_DrvDrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
- Draw a framed triangle.*

- bool [gslc\\_DrvDrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)  
*Draw a filled triangle.*
- bool [gslc\\_DrvDrawImage](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, [gslc\\_tsImgRef](#) slmgRef)  
*Copy all of source image to destination screen at specified coordinate.*
- void [gslc\\_DrvDrawMonoFromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)  
*Draw a monochrome bitmap from a memory array.*
- void [gslc\\_DrvDrawBmp24FromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)  
*Draw a color 24-bit depth bitmap from a memory array.*
- void [gslc\\_DrvDrawBkgnd](#) ([gslc\\_tsGui](#) \*pGui)  
*Copy the background image to destination screen.*
- bool [gslc\\_DrvRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)  
*Change rotation, automatically adapt touchscreen axes swap/flip.*
- uint16\_t [gslc\\_DrvAdaptColorToRaw](#) ([gslc\\_tsColor](#) nCol)

## Variables

- const char [GSLC\\_PMEM\\_ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM\\_ERRSTR\\_PXD\\_NULL](#) []

### 9.51.1 Detailed Description

GUISlice library (driver layer for M5stack)

### 9.51.2 Macro Definition Documentation

#### 9.51.2.1 DRV\_HAS\_DRAW\_BMP\_MEM

```
#define DRV_HAS_DRAW_BMP_MEM
```

Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)

#### 9.51.2.2 DRV\_HAS\_DRAW\_CIRCLE\_FILL

```
#define DRV_HAS_DRAW_CIRCLE_FILL
```

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.51.2.3 DRV\_HAS\_DRAW\_CIRCLE\_FRAME

```
#define DRV_HAS_DRAW_CIRCLE_FRAME
```

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.51.2.4 DRV\_HAS\_DRAW\_LINE

```
#define DRV_HAS_DRAW_LINE
```

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.51.2.5 DRV\_HAS\_DRAW\_POINT

```
#define DRV_HAS_DRAW_POINT
```

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.51.2.6 DRV\_HAS\_DRAW\_POINTS

```
#define DRV_HAS_DRAW_POINTS
```

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.51.2.7 DRV\_HAS\_DRAW\_RECT\_FILL

```
#define DRV_HAS_DRAW_RECT_FILL
```

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.51.2.8 DRV\_HAS\_DRAW\_RECT\_FRAME

```
#define DRV_HAS_DRAW_RECT_FRAME
```

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.51.2.9 DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

```
#define DRV_HAS_DRAW_RECT_ROUND_FILL
```

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.51.2.10 DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

```
#define DRV_HAS_DRAW_RECT_ROUND_FRAME
```

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.51.2.11 DRV\_HAS\_DRAW\_TEXT

```
#define DRV_HAS_DRAW_TEXT
```

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.51.2.12 DRV\_HAS\_DRAW\_TRI\_FILL

```
#define DRV_HAS_DRAW_TRI_FILL
```

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

#### 9.51.2.13 DRV\_HAS\_DRAW\_TRI\_FRAME

```
#define DRV_HAS_DRAW_TRI_FRAME
```

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

#### 9.51.2.14 DRV\_OVERRIDE\_TXT\_ALIGN

```
#define DRV_OVERRIDE_TXT_ALIGN
```

Driver provides text alignment.

### 9.51.3 Function Documentation

#### 9.51.3.1 `gslc_DrvAdaptColorToRaw()`

```
uint16_t gslc_DrvAdaptColorToRaw (
    gslc_tsColor nCol )
```

#### 9.51.3.2 `gslc_DrvDestruct()`

```
void gslc_DrvDestruct (
    gslc_tsGui * pGui )
```

Free up any members associated with the driver.

- Eg. renderers, windows, background surfaces, etc.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

##### Returns

none

#### 9.51.3.3 `gslc_DrvDrawBkgnd()`

```
void gslc_DrvDrawBkgnd (
    gslc_tsGui * pGui )
```

Copy the background image to destination screen.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

##### Returns

true if success, false if fail



## 9.51.3.4 gslc\_DrvDrawBmp24FromMem()

```
void gslc_DrvDrawBmp24FromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a color 24-bit depth bitmap from a memory array.

- Note that users must convert images from their native format (eg. BMP, PNG, etc.) into a C array. Please refer to the following guide for details: <https://github.com/ImpulseAdventure/GUIslice/wiki/Display-Images-from-FLASH>
- The converted file (c array) can then be included in the sketch.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	X coord for copy
in	<i>nDstY</i>	Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

## Returns

none

## 9.51.3.5 gslc\_DrvDrawFillCircle()

```
bool gslc_DrvDrawFillCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled circle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.51.3.6 gslc\_DrvDrawFillRect()**

```
bool gslc_DrvDrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.51.3.7 gslc\_DrvDrawFillRoundRect()**

```
bool gslc_DrvDrawFillRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

## 9.51.3.8 gslc\_DrvDrawFillTriangle()

```
bool gslc_DrvDrawFillTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a filled triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to fill

## Returns

true if success, false if error

## 9.51.3.9 gslc\_DrvDrawFrameCircle()

```
bool gslc_DrvDrawFrameCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed circle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

**9.51.3.10 gslc\_DrvDrawFrameRect()**

```
bool gslc_DrvDrawFrameRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a framed rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

**9.51.3.11 gslc\_DrvDrawFrameRoundRect()**

```
bool gslc_DrvDrawFrameRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

## 9.51.3.12 gslc\_DrvDrawFrameTriangle()

```
bool gslc_DrvDrawFrameTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a framed triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to frame

## Returns

true if success, false if error

## 9.51.3.13 gslc\_DrvDrawImage()

```
bool gslc_DrvDrawImage (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    gslc_tsImgRef sImgRef )
```

Copy all of source image to destination screen at specified coordinate.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>sImgRef</i>	Image reference

## Returns

true if success, false if fail

### 9.51.3.14 gslc\_DrvDrawLine()

```
bool gslc_DrvDrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw a line.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	Line start (X coordinate)
in	<i>nY0</i>	Line start (Y coordinate)
in	<i>nX1</i>	Line finish (X coordinate)
in	<i>nY1</i>	Line finish (Y coordinate)
in	<i>nCol</i>	Color RGB value to draw

#### Returns

true if success, false if error

### 9.51.3.15 gslc\_DrvDrawMonoFromMem()

```
void gslc_DrvDrawMonoFromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a monochrome bitmap from a memory array.

- Draw from the bitmap buffer using the foreground color defined in the header (unset bits are transparent)

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

**Returns**

none

**9.51.3.16 gslc\_DrvDrawPoint()**

```
bool gslc_DrvDrawPoint (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

**9.51.3.17 gslc\_DrvDrawPoints()**

```
bool gslc_DrvDrawPoints (
    gslc_tsGui * pGui,
    gslc_tsPt * asPt,
    uint16_t nNumPt,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>asPt</i>	Array of points to draw
in	$n \leftarrow$ <i>NumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

### 9.51.3.18 gslc\_DrvDrawTxt()

```
bool gslc_DrvDrawTxt (
    gslc_tsGui * pGui,
    int16_t nTxtX,
    int16_t nTxtY,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string at the given coordinate.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nTxtX</i>	X coordinate of top-left text string
in	<i>nTxtY</i>	Y coordinate of top-left text string
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	unused in m5stack, defaults to black

#### Returns

true if success, false if failure

### 9.51.3.19 gslc\_DrvDrawTxtAlign()

```
bool gslc_DrvDrawTxtAlign (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int8_t eTxtAlign,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string in a bounding box using the specified alignment.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X coordinate of top-left of bounding box
in	<i>nY0</i>	Y coordinate of top-left of bounding box
in	<i>nX1</i>	X coordinate of bot-right of bounding box
in	<i>nY1</i>	Y coordinate of bot-right of bounding box
in	<i>eTxtAlign</i>	Alignment mode]
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	unused in m5stack, defaults to black

## Returns

true if success, false if failure

## 9.51.3.20 gslc\_DrvFontAdd()

```
const void* gslc_DrvFontAdd (
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font from a resource and return pointer to it.

## Parameters

in	<i>eFontRefType</i>	Font reference type (GSLC_FONTREF_PTR for Arduino)
in	<i>pvFontRef</i>	Font reference pointer (Pointer to the GFXFont array)
in	<i>nFontSz</i>	Typeface size to use

## Returns

Void ptr to driver-specific font if load was successful, NULL otherwise

## 9.51.3.21 gslc\_DrvFontsDestruct()

```
void gslc_DrvFontsDestruct (
    gslc_tsGui * pGui )
```

Release all fonts defined in the GUI.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.51.3.22 gslc\_DrvGetDriverDisp()**

```
void* gslc_DrvGetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.51.3.23 gslc\_DrvGetDriverTouch()**

```
void* gslc_DrvGetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.51.3.24 gslc\_DrvGetNameDisp()**

```
const char* gslc_DrvGetNameDisp (  
    gslc_tsGui * pGui )
```

Get the display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.51.3.25 gslc\_DrvGetNameTouch()**

```
const char* gslc_DrvGetNameTouch (  
    gslc_tsGui * pGui )
```

Get the touch driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.51.3.26 gslc\_DrvGetTxtSize()**

```
bool gslc_DrvGetTxtSize (  
    gslc_tsGui * pGui,  
    gslc_tsFont * pFont,  
    const char * pStr,  
    gslc_teTxtFlags eTxtFlags,
```

```

    int16_t * pnTxtX,
    int16_t * pnTxtY,
    uint16_t * pnTxtSzW,
    uint16_t * pnTxtSzH )

```

Get the extent (width and height) of a text string.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pFont</i>	Ptr to Font structure
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
out	<i>pnTxtX</i>	Ptr to offset X of text
out	<i>pnTxtY</i>	Ptr to offset Y of text
out	<i>pnTxtSzW</i>	Ptr to width of text
out	<i>pnTxtSzH</i>	Ptr to height of text

#### Returns

true if success, false if failure

#### 9.51.3.27 gslc\_DrvImageDestruct()

```

void gslc_DrvImageDestruct (
    void * pvImg )

```

Release an image surface.

#### Parameters

in	<i>pvImg</i>	Void ptr to image
----	--------------	-------------------

#### Returns

none

#### 9.51.3.28 gslc\_DrvInit()

```

bool gslc_DrvInit (
    gslc_tsGui * pGui )

```

Initialize the SDL library.

- Performs clean startup workaround (if enabled)

- Configures video mode
- Initializes font support

PRE:

- The environment variables should be configured before calling `gslc_DrvInit()`. This can be done with `gslc_DrvInitEnv()` or manually in user function.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

true if success, false if fail

#### 9.51.3.29 gslc\_DrvInitTs()

```
bool gslc_DrvInitTs (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

#### Returns

true if successful

#### 9.51.3.30 gslc\_DrvLoadImage()

```
void* gslc_DrvLoadImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Load a bitmap (\*.bmp) and create a new image resource.

Transparency is enabled by `GSLC_BMP_TRANS_EN` through use of color (`GSLC_BMP_TRANS_RGB`).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

Image pointer (surface/texture) or NULL if error

**9.51.3.31 gslc\_DrvPageFlipNow()**

```
void gslc_DrvPageFlipNow (
    gslc_tsGui * pGui )
```

Force a page flip to occur.

This generally copies active screen surface to the display.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.51.3.32 gslc\_DrvRotate()**

```
bool gslc_DrvRotate (
    gslc_tsGui * pGui,
    uint8_t nRotation )
```

Change rotation, automatically adapt touchscreen axes swap/flip.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

**Returns**

true if successful

### 9.51.3.33 gslc\_DrvSetBkgndColor()

```
bool gslc_DrvSetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

#### Returns

true if success, false if fail

### 9.51.3.34 gslc\_DrvSetBkgndImage()

```
bool gslc_DrvSetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

#### Returns

true if success, false if fail

### 9.51.3.35 gslc\_DrvSetClipRect()

```
bool gslc_DrvSetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for future drawing updates.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Rectangular region to constrain edits

**Returns**

true if success, false if error

**9.51.3.36 gslc\_DrvSetElemImageGlow()**

```
bool gslc_DrvSetElemImageGlow (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's glow-state image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if error

**9.51.3.37 gslc\_DrvSetElemImageNorm()**

```
bool gslc_DrvSetElemImageNorm (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's normal-state image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference



### Returns

true if success, false if error

## 9.51.4 Variable Documentation

### 9.51.4.1 ERRSTR\_NULL

```
const char GSLC_PMEM ERRSTR_NULL[ ]
```

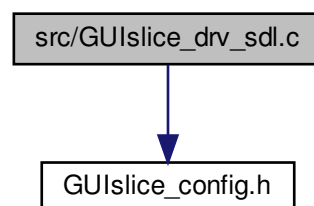
### 9.51.4.2 ERRSTR\_PXD\_NULL

```
const char GSLC_PMEM ERRSTR_PXD_NULL[ ]
```

## 9.52 src/GUISlice\_drv\_sdl.c File Reference

```
#include "GUISlice_config.h"
```

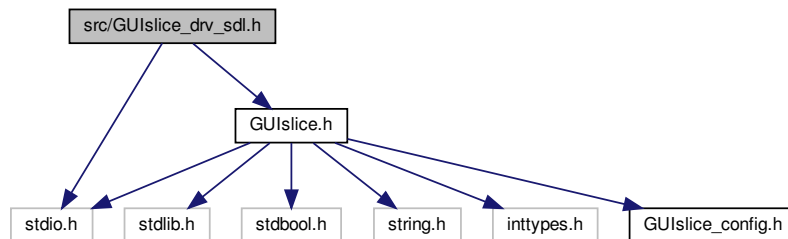
Include dependency graph for GUISlice\_drv\_sdl.c:



## 9.53 src/GUISlice\_drv\_sdl.h File Reference

GUISlice library (driver layer for LINUX / SDL)

```
#include "GUIslice.h"
#include <stdio.h>
Include dependency graph for GUIslice_drv_sdl.h:
```



## Data Structures

- struct [gslc\\_tsDriver](#)

## Macros

- #define [DRV\\_HAS\\_DRAW\\_POINT](#)  
Support [gslc\\_DrvDrawPoint\(\)](#)
- #define [DRV\\_OVERRIDE\\_TXT\\_ALIGN](#)  
Driver provides text alignment.

## Functions

- bool [gslc\\_DrvInit](#) ([gslc\\_tsGui](#) \*pGui)  
Initialize the SDL library.
- void [gslc\\_DrvDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
Free up any members associated with the driver.
- const char \* [gslc\\_DrvGetNameDisp](#) ([gslc\\_tsGui](#) \*pGui)  
Get the display driver name.
- const char \* [gslc\\_DrvGetNameTouch](#) ([gslc\\_tsGui](#) \*pGui)  
Get the touch driver name.
- void \* [gslc\\_DrvGetDriverDisp](#) ([gslc\\_tsGui](#) \*pGui)  
Get the native display driver instance.
- void \* [gslc\\_DrvGetDriverTouch](#) ([gslc\\_tsGui](#) \*pGui)  
Get the native touch driver instance.
- void \* [gslc\\_DrvLoadImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsImgRef](#) sImgRef)  
Load a bitmap (\*.bmp) and create a new image resource.
- bool [gslc\\_DrvSetBkgndImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsImgRef](#) sImgRef)  
Configure the background to use a bitmap image.
- bool [gslc\\_DrvSetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
Configure the background to use a solid color.
- bool [gslc\\_DrvSetElemImageNorm](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)  
Set an element's normal-state image.

- bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)  
*Set an element's glow-state image.*
- void [gslc\\_DrvImageDestruct](#) (void \*pvImg)  
*Release an image surface.*
- bool [gslc\\_DrvSetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for future drawing updates.*
- const void \* [gslc\\_DrvFontAdd](#) ([gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)  
*Load a font from a resource and return pointer to it.*
- void [gslc\\_DrvFontsDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Release all fonts defined in the GUI.*
- bool [gslc\\_DrvGetTxtSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, int16\_t \*pnTxtX, int16\_t \*pnTxtY, uint16\_t \*pnTxtSzW, uint16\_t \*pnTxtSzH)  
*Get the extent (width and height) of a text string.*
- bool [gslc\\_DrvDrawTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nTxtX, int16\_t nTxtY, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)  
*Draw a text string at the given coordinate.*
- void [gslc\\_DrvPageFlipNow](#) ([gslc\\_tsGui](#) \*pGui)  
*Force a page flip to occur.*
- bool [gslc\\_DrvDrawPoint](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawPoints](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*asPt, uint16\_t nNumPt, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a framed rectangle.*
- bool [gslc\\_DrvDrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)  
*Draw a filled rectangle.*
- bool [gslc\\_DrvDrawLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, [gslc\\_tsColor](#) nCol)  
*Draw a line.*
- bool [gslc\\_DrvDrawImage](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, [gslc\\_tsImgRef](#) sImgRef)  
*Copy all of source image to destination screen at specified coordinate.*
- void [gslc\\_DrvDrawBkgnd](#) ([gslc\\_tsGui](#) \*pGui)  
*Copy the background image to destination screen.*
- bool [gslc\\_DrvGetTouch](#) ([gslc\\_tsGui](#) \*pGui, int16\_t \*pnX, int16\_t \*pnY, uint16\_t \*pnPress, [gslc\\_teInputRawEvent](#) \*peInputEvent, int16\_t \*pnInputVal)  
*Get the last touch event from the SDL\_Event handler.*
- bool [gslc\\_DrvRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)  
*Change rotation, automatically adapt touchscreen axes swap/flip.*
- bool [gslc\\_DrvCleanStart](#) (const char \*sTTY)  
*Ensure SDL initializes cleanly to workaround possible issues if previous SDL application failed to close down gracefully.*
- void [gslc\\_DrvReportInfoPre](#) ()  
*Report driver debug info (before initialization)*
- void [gslc\\_DrvReportInfoPost](#) ()  
*Report driver debug info (after initialization)*
- SDL\_Rect [gslc\\_DrvAdaptRect](#) ([gslc\\_tsRect](#) rRect)  
*Translate a [gslc\\_tsRect](#) into an SDL\_Rect.*
- SDL\_Color [gslc\\_DrvAdaptColor](#) ([gslc\\_tsColor](#) sCol)  
*Translate a [gslc\\_tsColor](#) into an SDL\_Color.*
- bool [gslc\\_DrvInitTouch](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)  
*Perform any touchscreen-specific initialization.*

### 9.53.1 Detailed Description

GUIslice library (driver layer for LINUX / SDL)

### 9.53.2 Macro Definition Documentation

#### 9.53.2.1 DRV\_HAS\_DRAW\_POINT

```
#define DRV_HAS_DRAW_POINT
```

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.53.2.2 DRV\_OVERRIDE\_TXT\_ALIGN

```
#define DRV_OVERRIDE_TXT_ALIGN
```

Driver provides text alignment.

### 9.53.3 Function Documentation

#### 9.53.3.1 gslc\_DrvAdaptColor()

```
SDL_Color gslc_DrvAdaptColor (
    gslc\_tsColor sCol )
```

Translate a [gslc\\_tsColor](#) into an SDL\_Color.

##### Parameters

in	<i>sCol</i>	<a href="#">gslc_tsColor</a>
----	-------------	------------------------------

##### Returns

Converted SDL\_Color

9.53.3.2 `gslc_DrvAdaptRect()`

```
SDL_Rect gslc_DrvAdaptRect (
    gslc_tsRect rRect )
```

Translate a `gslc_tsRect` into an `SDL_Rect`.

## Parameters

in	<i>rRect</i>	<code>gslc_tsRect</code>
----	--------------	--------------------------

## Returns

Converted `SDL_Rect`

9.53.3.3 `gslc_DrvCleanStart()`

```
bool gslc_DrvCleanStart (
    const char * sTTY )
```

Ensure SDL initializes cleanly to workaround possible issues if previous SDL application failed to close down gracefully.

## Parameters

in	<i>sTTY</i>	Terminal device (eg. "/dev/tty0")
----	-------------	-----------------------------------

## Returns

true if success

9.53.3.4 `gslc_DrvDestruct()`

```
void gslc_DrvDestruct (
    gslc_tsGui * pGui )
```

Free up any members associated with the driver.

- Eg. renderers, windows, background surfaces, etc.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.53.3.5 gslc\_DrvDrawBkgnd()**

```
void gslc_DrvDrawBkgnd (
    gslc_tsGui * pGui )
```

Copy the background image to destination screen.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

true if success, false if fail

**9.53.3.6 gslc\_DrvDrawFillRect()**

```
bool gslc_DrvDrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.53.3.7 gslc\_DrvDrawFrameRect()**

```
bool gslc_DrvDrawFrameRect (
    gslc_tsGui * pGui,
```

```
gslc_tsRect rRect,  
gslc_tsColor nCol )
```

Draw a framed rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

**9.53.3.8 gslc\_DrvDrawImage()**

```
bool gslc_DrvDrawImage (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    gslc_tsImgRef sImgRef )
```

Copy all of source image to destination screen at specified coordinate.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if fail

**9.53.3.9 gslc\_DrvDrawLine()**

```
bool gslc_DrvDrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw a line.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	Line start (X coordinate)
in	<i>nY0</i>	Line start (Y coordinate)
in	<i>nX1</i>	Line finish (X coordinate)
in	<i>nY1</i>	Line finish (Y coordinate)
in	<i>nCol</i>	Color RGB value to draw

## Returns

true if success, false if error

## 9.53.3.10 gslc\_DrvDrawPoint()

```
bool gslc_DrvDrawPoint (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Draw a point.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point
in	<i>nCol</i>	Color RGB value to draw

## Returns

true if success, false if error

## 9.53.3.11 gslc\_DrvDrawPoints()

```
bool gslc_DrvDrawPoints (
    gslc_tsGui * pGui,
    gslc_tsPt * asPt,
    uint16_t nNumPt,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>asPt</i>	Array of points to draw
in	<i>n</i> ↔ <i>NumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

**9.53.3.12 gslc\_DrvDrawTxt()**

```
bool gslc_DrvDrawTxt (
    gslc_tsGui * pGui,
    int16_t nTxtX,
    int16_t nTxtY,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string at the given coordinate.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nTxtX</i>	X coordinate of top-left text string
in	<i>nTxtY</i>	Y coordinate of top-left text string
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	unused in SDL, defaults to black

**Returns**

true if success, false if failure

**9.53.3.13 gslc\_DrvFontAdd()**

```
const void* gslc_DrvFontAdd (
    gslc_teFontRefType eFontRefType,
```

```
const void * pvFontRef,
uint16_t nFontSz )
```

Load a font from a resource and return pointer to it.

#### Parameters

in	<i>eFontRefType</i>	Font reference type (GSLC_FONTREF_FNAME for SDL)
in	<i>pFontRef</i>	Font reference pointer (Pointer to the font filename)
in	<i>nFontSz</i>	Typeface size to use

#### Returns

Void ptr to driver-specific font if load was successful, NULL otherwise

#### 9.53.3.14 gslc\_DrvFontsDestruct()

```
void gslc_DrvFontsDestruct (
    gslc_tsGui * pGui )
```

Release all fonts defined in the GUI.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

none

#### 9.53.3.15 gslc\_DrvGetDriverDisp()

```
void* gslc_DrvGetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.53.3.16 gslc\_DrvGetDriverTouch()**

```
void* gslc_DrvGetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.53.3.17 gslc\_DrvGetNameDisp()**

```
const char* gslc_DrvGetNameDisp (
    gslc_tsGui * pGui )
```

Get the display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.53.3.18 gslc\_DrvGetNameTouch()**

```
const char* gslc_DrvGetNameTouch (
    gslc_tsGui * pGui )
```

Get the touch driver name.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

String containing driver name

## 9.53.3.19 gslc\_DrvGetTouch()

```
bool gslc_DrvGetTouch (
    gslc_tsGui * pGui,
    int16_t * pnX,
    int16_t * pnY,
    uint16_t * pnPress,
    gslc_teInputRawEvent * peInputEvent,
    int16_t * pnInputVal )
```

Get the last touch event from the SDL\_Event handler.

## Parameters

in	<i>pGui</i>	Pointer to GUI
out	<i>pnX</i>	Ptr to X coordinate of last touch event
out	<i>pnY</i>	Ptr to Y coordinate of last touch event
out	<i>pnPress</i>	Ptr to Pressure level of last touch event (0 for none, 1 for touch)
out	<i>peInputEvent</i>	Indication of event type
out	<i>pnInputVal</i>	Additional data for event type

## Returns

true if an event was detected or false otherwise

## 9.53.3.20 gslc\_DrvGetTxtSize()

```
bool gslc_DrvGetTxtSize (
    gslc_tsGui * pGui,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    int16_t * pnTxtX,
    int16_t * pnTxtY,
    uint16_t * pnTxtSzW,
    uint16_t * pnTxtSzH )
```

Get the extent (width and height) of a text string.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pFont</i>	Ptr to Font structure
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
out	<i>pnTxtX</i>	Ptr to offset X of text
out	<i>pnTxtY</i>	Ptr to offset Y of text
out	<i>pnTxtSzW</i>	Ptr to width of text
out	<i>pnTxtSzH</i>	Ptr to height of text

**Returns**

true if success, false if failure

**9.53.3.21 gslc\_DrvImageDestruct()**

```
void gslc_DrvImageDestruct (
    void * pvImg )
```

Release an image surface.

**Parameters**

in	<i>pvImg</i>	Void ptr to image
----	--------------	-------------------

**Returns**

none

**9.53.3.22 gslc\_DrvInit()**

```
bool gslc_DrvInit (
    gslc_tsGui * pGui )
```

Initialize the SDL library.

- Performs clean startup workaround (if enabled)
- Configures video mode
- Initializes font support

**PRE:**

- The environment variables should be configured before calling [gslc\\_DrvInit\(\)](#).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

true if success, false if fail

**9.53.3.23 gslc\_DrvInitTouch()**

```
bool gslc_DrvInitTouch (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

**Returns**

true if successful

**9.53.3.24 gslc\_DrvLoadImage()**

```
void* gslc_DrvLoadImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Load a bitmap (\*.bmp) and create a new image resource.

Transparency is enabled by GSLC\_BMP\_TRANS\_EN through use of color (GSLC\_BMP\_TRANS\_RGB).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

Image pointer (surface/texture/path) or NULL if error

#### 9.53.3.25 gslc\_DrvPageFlipNow()

```
void gslc_DrvPageFlipNow (
    gslc_tsGui * pGui )
```

Force a page flip to occur.

This generally copies active screen surface to the display.

##### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

##### Returns

none

#### 9.53.3.26 gslc\_DrvReportInfoPost()

```
void gslc_DrvReportInfoPost ( )
```

Report driver debug info (after initialization)

##### Returns

none

#### 9.53.3.27 gslc\_DrvReportInfoPre()

```
void gslc_DrvReportInfoPre ( )
```

Report driver debug info (before initialization)

##### Returns

none

#### 9.53.3.28 gslc\_DrvRotate()

```
bool gslc_DrvRotate (
    gslc_tsGui * pGui,
    uint8_t nRotation )
```

Change rotation, automatically adapt touchscreen axes swap/flip.



## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

## Returns

true if successful

## 9.53.3.29 gslc\_DrvSetBkgndColor()

```
bool gslc_DrvSetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

## Returns

true if success, false if fail

## 9.53.3.30 gslc\_DrvSetBkgndImage()

```
bool gslc_DrvSetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if fail

**9.53.3.31 gslc\_DrvSetClipRect()**

```
bool gslc_DrvSetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for future drawing updates.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Rectangular region to constrain edits

**Returns**

true if success, false if error

**9.53.3.32 gslc\_DrvSetElemImageGlow()**

```
bool gslc_DrvSetElemImageGlow (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's glow-state image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if error

**9.53.3.33 gslc\_DrvSetElemImageNorm()**

```
bool gslc_DrvSetElemImageNorm (
    gslc_tsGui * pGui,
```

```
gslc_tsElem * pElem,
gslc_tsImgRef sImgRef )
```

Set an element's normal-state image.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

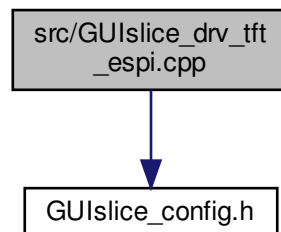
#### Returns

true if success, false if error

## 9.54 src/GUIslice\_drv\_tft\_espi.cpp File Reference

```
#include "GUIslice_config.h"
```

Include dependency graph for GUIslice\_drv\_tft\_espi.cpp:

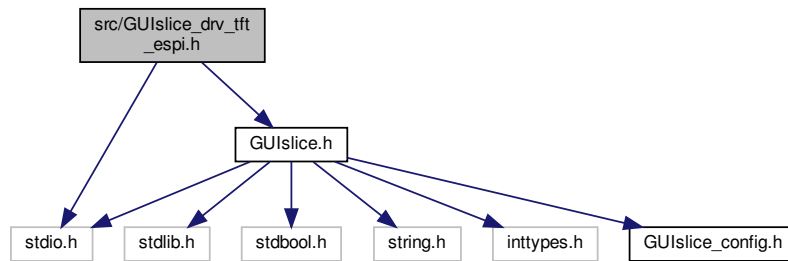


## 9.55 src/GUIslice\_drv\_tft\_espi.h File Reference

GUIslice library (driver layer for TFT-eSPI)

```
#include "GUIslice.h"
#include <stdio.h>
```

Include dependency graph for `GUIslice_drv_tft_espi.h`:



## Data Structures

- struct [gslc\\_tsDriver](#)

## Macros

- `#define` [GSLC\\_SPIFFS\\_EN](#)
- `#define` [DRV\\_HAS\\_DRAW\\_POINT](#)  
Support [gslc\\_DrvDrawPoint\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_POINTS](#)  
Support [gslc\\_DrvDrawPoints\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_LINE](#)  
Support [gslc\\_DrvDrawLine\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_RECT\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameRect\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_RECT\\_FILL](#)  
Support [gslc\\_DrvDrawFillRect\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_RECT\\_ROUND\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_RECT\\_ROUND\\_FILL](#)  
Support [gslc\\_DrvDrawFillRoundRect\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_CIRCLE\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameCircle\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_CIRCLE\\_FILL](#)  
Support [gslc\\_DrvDrawFillCircle\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_TRI\\_FRAME](#)  
Support [gslc\\_DrvDrawFrameTriangle\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_TRI\\_FILL](#)  
Support [gslc\\_DrvDrawFillTriangle\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_TEXT](#)  
Support [gslc\\_DrvDrawTxt\(\)](#)
- `#define` [DRV\\_HAS\\_DRAW\\_BMP\\_MEM](#)  
Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)
- `#define` [DRV\\_OVERRIDE\\_TXT\\_ALIGN](#)  
Driver provides text alignment.

## Functions

- bool [gslc\\_DrvInit](#) ([gslc\\_tsGui](#) \*pGui)  
*Initialize the SDL library.*
- bool [gslc\\_DrvInitTs](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)  
*Perform any touchscreen-specific initialization.*
- void [gslc\\_DrvDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Free up any members associated with the driver.*
- const char \* [gslc\\_DrvGetNameDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the display driver name.*
- const char \* [gslc\\_DrvGetNameTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the touch driver name.*
- void \* [gslc\\_DrvGetDriverDisp](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native display driver instance.*
- void \* [gslc\\_DrvGetDriverTouch](#) ([gslc\\_tsGui](#) \*pGui)  
*Get the native touch driver instance.*
- void \* [gslc\\_DrvLoadImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_slmgRef](#) slmgRef)  
*Load a bitmap (\*.bmp) and create a new image resource.*
- bool [gslc\\_DrvSetBkgndImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_slmgRef](#) slmgRef)  
*Configure the background to use a bitmap image.*
- bool [gslc\\_DrvSetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)  
*Configure the background to use a solid color.*
- bool [gslc\\_DrvSetElemImageNorm](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_slmgRef](#) slmgRef)  
*Set an element's normal-state image.*
- bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_slmgRef](#) slmgRef)  
*Set an element's glow-state image.*
- void [gslc\\_DrvImageDestruct](#) (void \*pvImg)  
*Release an image surface.*
- bool [gslc\\_DrvSetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for future drawing updates.*
- const void \* [gslc\\_DrvFontAdd](#) ([gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)  
*Load a font from a resource and return pointer to it.*
- void [gslc\\_DrvFontsDestruct](#) ([gslc\\_tsGui](#) \*pGui)  
*Release all fonts defined in the GUI.*
- bool [gslc\\_DrvGetTxtSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxt↵Flags, int16\_t \*pnTxtX, int16\_t \*pnTxtY, uint16\_t \*pnTxtSzW, uint16\_t \*pnTxtSzH)  
*Get the extent (width and height) of a text string.*
- bool [gslc\\_DrvDrawTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nTxtX, int16\_t nTxtY, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)  
*Draw a text string at the given coordinate.*
- bool [gslc\\_DrvDrawTxtAlign](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int8\_t e↵TxtAlign, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)  
*Draw a text string in a bounding box using the specified alignment.*
- void [gslc\\_DrvPageFlipNow](#) ([gslc\\_tsGui](#) \*pGui)  
*Force a page flip to occur.*
- bool [gslc\\_DrvDrawPoint](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawPoints](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*asPt, uint16\_t nNumPt, [gslc\\_tsColor](#) nCol)  
*Draw a point.*
- bool [gslc\\_DrvDrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)

*Draw a framed rectangle.*

- bool [gslc\\_DrvDrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)

*Draw a filled rectangle.*

- bool [gslc\\_DrvDrawFrameRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)

*Draw a framed rounded rectangle.*

- bool [gslc\\_DrvDrawFillRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)

*Draw a filled rounded rectangle.*

- bool [gslc\\_DrvDrawLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, [gslc\\_tsColor](#) nCol)

*Draw a line.*

- bool [gslc\\_DrvDrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)

*Draw a framed circle.*

- bool [gslc\\_DrvDrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)

*Draw a filled circle.*

- bool [gslc\\_DrvDrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)

*Draw a framed triangle.*

- bool [gslc\\_DrvDrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)

*Draw a filled triangle.*

- bool [gslc\\_DrvDrawImage](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, [gslc\\_tsImgRef](#) sImgRef)

*Copy all of source image to destination screen at specified coordinate.*

- void [gslc\\_DrvDrawMonoFromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)

*Draw a monochrome bitmap from a memory array.*

- void [gslc\\_DrvDrawBmp24FromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)

*Draw a color 24-bit depth bitmap from a memory array.*

- void [gslc\\_DrvDrawBkgnd](#) ([gslc\\_tsGui](#) \*pGui)

*Copy the background image to destination screen.*

- bool [gslc\\_DrvRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)

*Change rotation, automatically adapt touchscreen axes swap/flip.*

- uint16\_t [gslc\\_DrvAdaptColorToRaw](#) ([gslc\\_tsColor](#) nCol)

### 9.55.1 Detailed Description

GUIslice library (driver layer for TFT-eSPI)

### 9.55.2 Macro Definition Documentation

#### 9.55.2.1 DRV\_HAS\_DRAW\_BMP\_MEM

```
#define DRV_HAS_DRAW_BMP_MEM
```

Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)

#### 9.55.2.2 DRV\_HAS\_DRAW\_CIRCLE\_FILL

```
#define DRV_HAS_DRAW_CIRCLE_FILL
```

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.55.2.3 DRV\_HAS\_DRAW\_CIRCLE\_FRAME

```
#define DRV_HAS_DRAW_CIRCLE_FRAME
```

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.55.2.4 DRV\_HAS\_DRAW\_LINE

```
#define DRV_HAS_DRAW_LINE
```

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.55.2.5 DRV\_HAS\_DRAW\_POINT

```
#define DRV_HAS_DRAW_POINT
```

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.55.2.6 DRV\_HAS\_DRAW\_POINTS

```
#define DRV_HAS_DRAW_POINTS
```

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.55.2.7 DRV\_HAS\_DRAW\_RECT\_FILL

```
#define DRV_HAS_DRAW_RECT_FILL
```

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.55.2.8 DRV\_HAS\_DRAW\_RECT\_FRAME

```
#define DRV_HAS_DRAW_RECT_FRAME
```

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.55.2.9 DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

```
#define DRV_HAS_DRAW_RECT_ROUND_FILL
```

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.55.2.10 DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

```
#define DRV_HAS_DRAW_RECT_ROUND_FRAME
```

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.55.2.11 DRV\_HAS\_DRAW\_TEXT

```
#define DRV_HAS_DRAW_TEXT
```

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.55.2.12 DRV\_HAS\_DRAW\_TRI\_FILL

```
#define DRV_HAS_DRAW_TRI_FILL
```

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

#### 9.55.2.13 DRV\_HAS\_DRAW\_TRI\_FRAME

```
#define DRV_HAS_DRAW_TRI_FRAME
```

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)



#### 9.55.2.14 DRV\_OVERRIDE\_TXT\_ALIGN

```
#define DRV_OVERRIDE_TXT_ALIGN
```

Driver provides text alignment.

#### 9.55.2.15 GSLC\_SPIFFS\_EN

```
#define GSLC_SPIFFS_EN
```

### 9.55.3 Function Documentation

#### 9.55.3.1 gslc\_DrvAdaptColorToRaw()

```
uint16_t gslc_DrvAdaptColorToRaw (  
    gslc_tsColor nCol )
```

#### 9.55.3.2 gslc\_DrvDestruct()

```
void gslc_DrvDestruct (  
    gslc_tsGui * pGui )
```

Free up any members associated with the driver.

- Eg. renderers, windows, background surfaces, etc.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

none

#### 9.55.3.3 gslc\_DrvDrawBkgnd()

```
void gslc_DrvDrawBkgnd (  
    gslc_tsGui * pGui )
```

Copy the background image to destination screen.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

true if success, false if fail

## 9.55.3.4 gslc\_DrvDrawBmp24FromMem()

```
void gslc_DrvDrawBmp24FromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a color 24-bit depth bitmap from a memory array.

- Note that users must convert images from their native format (eg. BMP, PNG, etc.) into a C array. Please refer to the following guide for details: <https://github.com/ImpulseAdventure/GUISlice/wiki/Display-Images-from-FLASH>
- The converted file (c array) can then be included in the sketch.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	X coord for copy
in	<i>nDstY</i>	Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

## Returns

none

## 9.55.3.5 gslc\_DrvDrawFillCircle()

```
bool gslc_DrvDrawFillCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled circle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.55.3.6 gslc\_DrvDrawFillRect()**

```
bool gslc_DrvDrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.55.3.7 gslc\_DrvDrawFillRoundRect()**

```
bool gslc_DrvDrawFillRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.55.3.8 gslc\_DrvDrawFillTriangle()**

```
bool gslc_DrvDrawFillTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a filled triangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.55.3.9 gslc\_DrvDrawFrameCircle()**

```
bool gslc_DrvDrawFrameCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed circle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

**9.55.3.10 gslc\_DrvDrawFrameRect()**

```
bool gslc_DrvDrawFrameRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a framed rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

**9.55.3.11 gslc\_DrvDrawFrameRoundRect()**

```
bool gslc_DrvDrawFrameRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed rounded rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to frame

**Returns**

true if success, false if error

## 9.55.3.12 gslc\_DrvDrawFrameTriangle()

```
bool gslc_DrvDrawFrameTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a framed triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to frame

## Returns

true if success, false if error

## 9.55.3.13 gslc\_DrvDrawImage()

```
bool gslc_DrvDrawImage (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    gslc_tsImgRef sImgRef )
```

Copy all of source image to destination screen at specified coordinate.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>sImgRef</i>	Image reference

## Returns

true if success, false if fail

### 9.55.3.14 gslc\_DrvDrawLine()

```
bool gslc_DrvDrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw a line.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	Line start (X coordinate)
in	<i>nY0</i>	Line start (Y coordinate)
in	<i>nX1</i>	Line finish (X coordinate)
in	<i>nY1</i>	Line finish (Y coordinate)
in	<i>nCol</i>	Color RGB value to draw

#### Returns

true if success, false if error

### 9.55.3.15 gslc\_DrvDrawMonoFromMem()

```
void gslc_DrvDrawMonoFromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a monochrome bitmap from a memory array.

- Draw from the bitmap buffer using the foreground color defined in the header (unset bits are transparent)

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise



**Returns**

none

**9.55.3.16 gslc\_DrvDrawPoint()**

```
bool gslc_DrvDrawPoint (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

**9.55.3.17 gslc\_DrvDrawPoints()**

```
bool gslc_DrvDrawPoints (
    gslc_tsGui * pGui,
    gslc_tsPt * asPt,
    uint16_t nNumPt,
    gslc_tsColor nCol )
```

Draw a point.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>asPt</i>	Array of points to draw
in	<i>n</i> $\leftrightarrow$ <i>NumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

**Returns**

true if success, false if error

### 9.55.3.18 gslc\_DrvDrawTxt()

```
bool gslc_DrvDrawTxt (
    gslc_tsGui * pGui,
    int16_t nTxtX,
    int16_t nTxtY,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string at the given coordinate.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nTxtX</i>	X coordinate of top-left text string
in	<i>nTxtY</i>	Y coordinate of top-left text string
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	Color of Background for antialias blending

#### Returns

true if success, false if failure

### 9.55.3.19 gslc\_DrvDrawTxtAlign()

```
bool gslc_DrvDrawTxtAlign (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int8_t eTxtAlign,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string in a bounding box using the specified alignment.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X coordinate of top-left of bounding box
in	<i>nY0</i>	Y coordinate of top-left of bounding box
in	<i>nX1</i>	X coordinate of bot-right of bounding box
in	<i>nY1</i>	Y coordinate of bot-right of bounding box
in	<i>eTxtAlign</i>	Alignment mode]
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	Color of Background for antialias blending

## Returns

true if success, false if failure

## 9.55.3.20 gslc\_DrvFontAdd()

```
const void* gslc_DrvFontAdd (
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font from a resource and return pointer to it.

## Parameters

in	<i>eFontRefType</i>	Font reference type: <ul style="list-style-type: none"> <li>• GSLC_FONTREF_PTR for Standard TFT_eSPI Fonts</li> <li>• GSLC_FONTREF_FNAME for antialiased Font in SPIFFS</li> </ul>
in	<i>pvFontRef</i>	Font reference pointer / SPIFFS font filename without ext.
in	<i>nFontSz</i>	Typeface size to use, ignored for SPIFFS font

## Returns

Void ptr to driver-specific font if load was successful, NULL otherwise

## 9.55.3.21 gslc\_DrvFontsDestruct()

```
void gslc_DrvFontsDestruct (
    gslc_tsGui * pGui )
```

Release all fonts defined in the GUI.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.55.3.22 gslc\_DrvGetDriverDisp()**

```
void* gslc_DrvGetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.55.3.23 gslc\_DrvGetDriverTouch()**

```
void* gslc_DrvGetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.55.3.24 gslc\_DrvGetNameDisp()**

```
const char* gslc_DrvGetNameDisp (
    gslc_tsGui * pGui )
```

Get the display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.55.3.25 gslc\_DrvGetNameTouch()**

```
const char* gslc_DrvGetNameTouch (
    gslc_tsGui * pGui )
```

Get the touch driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.55.3.26 gslc\_DrvGetTxtSize()**

```
bool gslc_DrvGetTxtSize (
    gslc_tsGui * pGui,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
```

```

    int16_t * pnTxtX,
    int16_t * pnTxtY,
    uint16_t * pnTxtSzW,
    uint16_t * pnTxtSzH )

```

Get the extent (width and height) of a text string.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pFont</i>	Ptr to Font structure
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
out	<i>pnTxtX</i>	Ptr to offset X of text
out	<i>pnTxtY</i>	Ptr to offset Y of text
out	<i>pnTxtSzW</i>	Ptr to width of text
out	<i>pnTxtSzH</i>	Ptr to height of text

#### Returns

true if success, false if failure

#### 9.55.3.27 gslc\_DrvImageDestruct()

```

void gslc_DrvImageDestruct (
    void * pvImg )

```

Release an image surface.

#### Parameters

in	<i>pvImg</i>	Void ptr to image
----	--------------	-------------------

#### Returns

none

#### 9.55.3.28 gslc\_DrvInit()

```

bool gslc_DrvInit (
    gslc_tsGui * pGui )

```

Initialize the SDL library.

- Performs clean startup workaround (if enabled)

- Configures video mode
- Initializes font support

PRE:

- The environment variables should be configured before calling `gslc_DrvInit()`. This can be done with `gslc_DrvInitEnv()` or manually in user function.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

true if success, false if fail

#### 9.55.3.29 gslc\_DrvInitTs()

```
bool gslc_DrvInitTs (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

#### Returns

true if successful

#### 9.55.3.30 gslc\_DrvLoadImage()

```
void* gslc_DrvLoadImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Load a bitmap (\*.bmp) and create a new image resource.

Transparency is enabled by `GSLC_BMP_TRANS_EN` through use of color (`GSLC_BMP_TRANS_RGB`).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

Image pointer (surface/texture) or NULL if error

**9.55.3.31 gslc\_DrvPageFlipNow()**

```
void gslc_DrvPageFlipNow (
    gslc_tsGui * pGui )
```

Force a page flip to occur.

This generally copies active screen surface to the display.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.55.3.32 gslc\_DrvRotate()**

```
bool gslc_DrvRotate (
    gslc_tsGui * pGui,
    uint8_t nRotation )
```

Change rotation, automatically adapt touchscreen axes swap/flip.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

**Returns**

true if successful



#### 9.55.3.33 gslc\_DrvSetBkgndColor()

```
bool gslc_DrvSetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

##### Returns

true if success, false if fail

#### 9.55.3.34 gslc\_DrvSetBkgndImage()

```
bool gslc_DrvSetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

##### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

##### Returns

true if success, false if fail

#### 9.55.3.35 gslc\_DrvSetClipRect()

```
bool gslc_DrvSetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for future drawing updates.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Rectangular region to constrain edits

**Returns**

true if success, false if error

**9.55.3.36 gslc\_DrvSetElemImageGlow()**

```
bool gslc_DrvSetElemImageGlow (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's glow-state image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if error

**9.55.3.37 gslc\_DrvSetElemImageNorm()**

```
bool gslc_DrvSetElemImageNorm (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's normal-state image.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

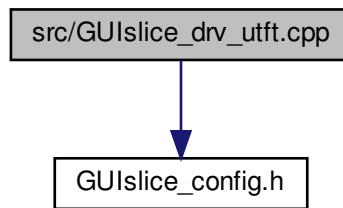
**Returns**

true if success, false if error

**9.56 src/GUISlice\_drv\_utft.cpp File Reference**

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_utft.cpp:

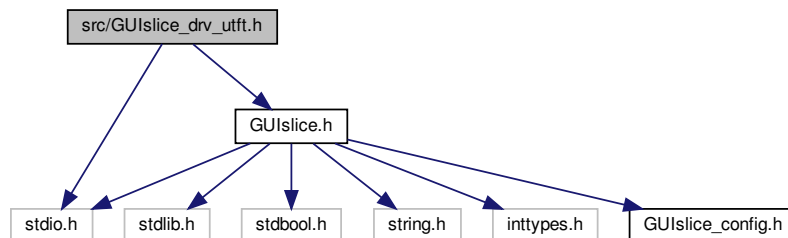
**9.57 src/GUISlice\_drv\_utft.h File Reference**

GUISlice library (driver layer for UTFT)

```
#include "GUISlice.h"
```

```
#include <stdio.h>
```

Include dependency graph for GUISlice\_drv\_utft.h:

**Data Structures**

- struct [gslc\\_tsDriver](#)

## Macros

- `#define DRV_HAS_DRAW_POINT`  
Support `gslc_DrvDrawPoint()`
- `#define DRV_HAS_DRAW_POINTS`  
Support `gslc_DrvDrawPoints()`
- `#define DRV_HAS_DRAW_LINE`  
Support `gslc_DrvDrawLine()`
- `#define DRV_HAS_DRAW_RECT_FRAME`  
Support `gslc_DrvDrawFrameRect()`
- `#define DRV_HAS_DRAW_RECT_FILL`  
Support `gslc_DrvDrawFillRect()`
- `#define DRV_HAS_DRAW_RECT_ROUND_FRAME`  
Support `gslc_DrvDrawFrameRoundRect()`
- `#define DRV_HAS_DRAW_RECT_ROUND_FILL`  
Support `gslc_DrvDrawFillRoundRect()`
- `#define DRV_HAS_DRAW_CIRCLE_FRAME`  
Support `gslc_DrvDrawFrameCircle()`
- `#define DRV_HAS_DRAW_CIRCLE_FILL`  
Support `gslc_DrvDrawFillCircle()`
- `#define DRV_HAS_DRAW_TRI_FRAME`  
Support `gslc_DrvDrawFrameTriangle()`
- `#define DRV_HAS_DRAW_TRI_FILL`  
Support `gslc_DrvDrawFillTriangle()`
- `#define DRV_HAS_DRAW_TEXT`  
Support `gslc_DrvDrawTxt()`
- `#define DRV_HAS_DRAW_BMP_MEM`  
Support `gslc_DrvDrawBmp24FromMem()`
- `#define DRV_OVERRIDE_TXT_ALIGN`  
Driver provides text alignment.

## Functions

- `bool gslc_DrvInit (gslc_tsGui *pGui)`  
Initialize the SDL library.
- `bool gslc_DrvInitTs (gslc_tsGui *pGui, const char *acDev)`  
Perform any touchscreen-specific initialization.
- `void gslc_DrvDestruct (gslc_tsGui *pGui)`  
Free up any members associated with the driver.
- `const char * gslc_DrvGetNameDisp (gslc_tsGui *pGui)`  
Get the display driver name.
- `const char * gslc_DrvGetNameTouch (gslc_tsGui *pGui)`  
Get the touch driver name.
- `void * gslc_DrvGetDriverDisp (gslc_tsGui *pGui)`  
Get the native display driver instance.
- `void * gslc_DrvGetDriverTouch (gslc_tsGui *pGui)`  
Get the native touch driver instance.
- `void * gslc_DrvLoadImage (gslc_tsGui *pGui, gslc_tsImgRef sImgRef)`  
Load a bitmap (\*.bmp) and create a new image resource.
- `bool gslc_DrvSetBkgndImage (gslc_tsGui *pGui, gslc_tsImgRef sImgRef)`

- Configure the background to use a bitmap image.*

  - bool [gslc\\_DrvSetBkgndColor](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsColor](#) nCol)
- Configure the background to use a solid color.*

  - bool [gslc\\_DrvSetElemImageNorm](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)
- Set an element's normal-state image.*

  - bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) sImgRef)
- Set an element's glow-state image.*

  - void [gslc\\_DrvImageDestruct](#) (void \*pvImg)
- Release an image surface.*

  - bool [gslc\\_DrvSetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)
- Set the clipping rectangle for future drawing updates.*

  - const void \* [gslc\\_DrvFontAdd](#) ([gslc\\_teFontRefType](#) eFontRefType, const void \*pvFontRef, uint16\_t nFontSz)
- Load a font from a resource and return pointer to it.*

  - void [gslc\\_DrvFontsDestruct](#) ([gslc\\_tsGui](#) \*pGui)
- Release all fonts defined in the GUI.*

  - bool [gslc\\_DrvGetTxtSize](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, int16\_t \*pnTxtX, int16\_t \*pnTxtY, uint16\_t \*pnTxtSzW, uint16\_t \*pnTxtSzH)
- Get the extent (width and height) of a text string.*

  - bool [gslc\\_DrvDrawTxt](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nTxtX, int16\_t nTxtY, [gslc\\_tsFont](#) \*pFont, const char \*pStr, [gslc\\_teTxtFlags](#) eTxtFlags, [gslc\\_tsColor](#) colTxt, [gslc\\_tsColor](#) colBg)
- Draw a text string at the given coordinate.*

  - void [gslc\\_DrvPageFlipNow](#) ([gslc\\_tsGui](#) \*pGui)
- Force a page flip to occur.*

  - bool [gslc\\_DrvDrawPoint](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX, int16\_t nY, [gslc\\_tsColor](#) nCol)
- Draw a point.*

  - bool [gslc\\_DrvDrawPoints](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPt](#) \*asPt, uint16\_t nNumPt, [gslc\\_tsColor](#) nCol)
- Draw a point.*

  - bool [gslc\\_DrvDrawFrameRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)
- Draw a framed rectangle.*

  - bool [gslc\\_DrvDrawFillRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, [gslc\\_tsColor](#) nCol)
- Draw a filled rectangle.*

  - bool [gslc\\_DrvDrawFrameRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a framed rounded rectangle.*

  - bool [gslc\\_DrvDrawFillRoundRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) rRect, int16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a filled rounded rectangle.*

  - bool [gslc\\_DrvDrawLine](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, [gslc\\_tsColor](#) nCol)
- Draw a line.*

  - bool [gslc\\_DrvDrawFrameCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a framed circle.*

  - bool [gslc\\_DrvDrawFillCircle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nMidX, int16\_t nMidY, uint16\_t nRadius, [gslc\\_tsColor](#) nCol)
- Draw a filled circle.*

  - bool [gslc\\_DrvDrawFrameTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
- Draw a framed triangle.*

  - bool [gslc\\_DrvDrawFillTriangle](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nX0, int16\_t nY0, int16\_t nX1, int16\_t nY1, int16\_t nX2, int16\_t nY2, [gslc\\_tsColor](#) nCol)
- Draw a filled triangle.*

  - bool [gslc\\_DrvDrawImage](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, [gslc\\_tsImgRef](#) sImgRef)

*Copy all of source image to destination screen at specified coordinate.*

- void [gslc\\_DrvDrawMonoFromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)

*Draw a monochrome bitmap from a memory array.*

- void [gslc\\_DrvDrawBmp24FromMem](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nDstX, int16\_t nDstY, const unsigned char \*pBitmap, bool bProgMem)

*Draw a color 24-bit depth bitmap from a memory array.*

- void [gslc\\_DrvDrawBkgnd](#) ([gslc\\_tsGui](#) \*pGui)

*Copy the background image to destination screen.*

- bool [gslc\\_DrvInitTouch](#) ([gslc\\_tsGui](#) \*pGui, const char \*acDev)

*Perform any touchscreen-specific initialization.*

- bool [gslc\\_DrvGetTouch](#) ([gslc\\_tsGui](#) \*pGui, int16\_t \*pnX, int16\_t \*pnY, uint16\_t \*pnPress, [gslc\\_tsInputRawEvent](#) \*peInputEvent, int16\_t \*pnInputVal)

*Get the last touch event from the internal touch handler.*

- bool [gslc\\_DrvRotate](#) ([gslc\\_tsGui](#) \*pGui, uint8\_t nRotation)

*Change rotation, automatically adapt touchscreen axes swap/flip.*

- uint16\_t [gslc\\_DrvAdaptColorToRaw](#) ([gslc\\_tsColor](#) nCol)

### 9.57.1 Detailed Description

GUIslice library (driver layer for UTFT)

### 9.57.2 Macro Definition Documentation

#### 9.57.2.1 DRV\_HAS\_DRAW\_BMP\_MEM

```
#define DRV_HAS_DRAW_BMP_MEM
```

Support [gslc\\_DrvDrawBmp24FromMem\(\)](#)

#### 9.57.2.2 DRV\_HAS\_DRAW\_CIRCLE\_FILL

```
#define DRV_HAS_DRAW_CIRCLE_FILL
```

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.57.2.3 DRV\_HAS\_DRAW\_CIRCLE\_FRAME

```
#define DRV_HAS_DRAW_CIRCLE_FRAME
```

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.57.2.4 DRV\_HAS\_DRAW\_LINE

```
#define DRV_HAS_DRAW_LINE
```

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.57.2.5 DRV\_HAS\_DRAW\_POINT

```
#define DRV_HAS_DRAW_POINT
```

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.57.2.6 DRV\_HAS\_DRAW\_POINTS

```
#define DRV_HAS_DRAW_POINTS
```

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.57.2.7 DRV\_HAS\_DRAW\_RECT\_FILL

```
#define DRV_HAS_DRAW_RECT_FILL
```

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.57.2.8 DRV\_HAS\_DRAW\_RECT\_FRAME

```
#define DRV_HAS_DRAW_RECT_FRAME
```

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.57.2.9 DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

```
#define DRV_HAS_DRAW_RECT_ROUND_FILL
```

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.57.2.10 DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

```
#define DRV_HAS_DRAW_RECT_ROUND_FRAME
```

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.57.2.11 DRV\_HAS\_DRAW\_TEXT

```
#define DRV_HAS_DRAW_TEXT
```

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.57.2.12 DRV\_HAS\_DRAW\_TRI\_FILL

```
#define DRV_HAS_DRAW_TRI_FILL
```

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

#### 9.57.2.13 DRV\_HAS\_DRAW\_TRI\_FRAME

```
#define DRV_HAS_DRAW_TRI_FRAME
```

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

#### 9.57.2.14 DRV\_OVERRIDE\_TXT\_ALIGN

```
#define DRV_OVERRIDE_TXT_ALIGN
```

Driver provides text alignment.

### 9.57.3 Function Documentation

#### 9.57.3.1 gslc\_DrvAdaptColorToRaw()

```
uint16_t gslc_DrvAdaptColorToRaw (
    gslc\_tsColor nCol )
```

#### 9.57.3.2 gslc\_DrvDestruct()

```
void gslc_DrvDestruct (
    gslc\_tsGui * pGui )
```

Free up any members associated with the driver.

- Eg. renderers, windows, background surfaces, etc.



## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

none

## 9.57.3.3 gslc\_DrvDrawBkgnd()

```
void gslc_DrvDrawBkgnd (
    gslc_tsGui * pGui )
```

Copy the background image to destination screen.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

true if success, false if fail

## 9.57.3.4 gslc\_DrvDrawBmp24FromMem()

```
void gslc_DrvDrawBmp24FromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a color 24-bit depth bitmap from a memory array.

- Note that users must convert images from their native format (eg. BMP, PNG, etc.) into a C array. Please refer to the following guide for details: <https://github.com/ImpulseAdventure/GUISlice/wiki/Display-Images-from-FLASH>
- The converted file (c array) can then be included in the sketch.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	X coord for copy
in	<i>nDstY</i>	Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

**Returns**

none

**9.57.3.5 gslc\_DrvDrawFillCircle()**

```
bool gslc_DrvDrawFillCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled circle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

**9.57.3.6 gslc\_DrvDrawFillRect()**

```
bool gslc_DrvDrawFillRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a filled rectangle.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nCol</i>	Color RGB value to fill

**Returns**

true if success, false if error

## 9.57.3.7 gslc\_DrvDrawFillRoundRect()

```
bool gslc_DrvDrawFillRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a filled rounded rectangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to fill
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to fill

## Returns

true if success, false if error

## 9.57.3.8 gslc\_DrvDrawFillTriangle()

```
bool gslc_DrvDrawFillTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a filled triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to fill

## Returns

true if success, false if error

### 9.57.3.9 gslc\_DrvDrawFrameCircle()

```
bool gslc_DrvDrawFrameCircle (
    gslc_tsGui * pGui,
    int16_t nMidX,
    int16_t nMidY,
    uint16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed circle.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nMidX</i>	Center of circle (X coordinate)
in	<i>nMidY</i>	Center of circle (Y coordinate)
in	<i>nRadius</i>	Radius of circle
in	<i>nCol</i>	Color RGB value to frame

#### Returns

true if success, false if error

### 9.57.3.10 gslc\_DrvDrawFrameRect()

```
bool gslc_DrvDrawFrameRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    gslc_tsColor nCol )
```

Draw a framed rectangle.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nCol</i>	Color RGB value to frame

#### Returns

true if success, false if error

## 9.57.3.11 gslc\_DrvDrawFrameRoundRect()

```
bool gslc_DrvDrawFrameRoundRect (
    gslc_tsGui * pGui,
    gslc_tsRect rRect,
    int16_t nRadius,
    gslc_tsColor nCol )
```

Draw a framed rounded rectangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>rRect</i>	Rectangular region to frame
in	<i>nRadius</i>	Radius for rounded corners
in	<i>nCol</i>	Color RGB value to frame

## Returns

true if success, false if error

## 9.57.3.12 gslc\_DrvDrawFrameTriangle()

```
bool gslc_DrvDrawFrameTriangle (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    int16_t nX2,
    int16_t nY2,
    gslc_tsColor nCol )
```

Draw a framed triangle.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	X Coordinate #1
in	<i>nY0</i>	Y Coordinate #1
in	<i>nX1</i>	X Coordinate #2
in	<i>nY1</i>	Y Coordinate #2
in	<i>nX2</i>	X Coordinate #3
in	<i>nY2</i>	Y Coordinate #3
in	<i>nCol</i>	Color RGB value to frame

## Returns

true if success, false if error

### 9.57.3.13 gslc\_DrvDrawImage()

```
bool gslc_DrvDrawImage (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    gslc_tsImgRef sImgRef )
```

Copy all of source image to destination screen at specified coordinate.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>sImgRef</i>	Image reference

#### Returns

true if success, false if fail

### 9.57.3.14 gslc\_DrvDrawLine()

```
bool gslc_DrvDrawLine (
    gslc_tsGui * pGui,
    int16_t nX0,
    int16_t nY0,
    int16_t nX1,
    int16_t nY1,
    gslc_tsColor nCol )
```

Draw a line.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX0</i>	Line start (X coordinate)
in	<i>nY0</i>	Line start (Y coordinate)
in	<i>nX1</i>	Line finish (X coordinate)
in	<i>nY1</i>	Line finish (Y coordinate)
in	<i>nCol</i>	Color RGB value to draw

#### Returns

true if success, false if error

## 9.57.3.15 gslc\_DrvDrawMonoFromMem()

```
void gslc_DrvDrawMonoFromMem (
    gslc_tsGui * pGui,
    int16_t nDstX,
    int16_t nDstY,
    const unsigned char * pBitmap,
    bool bProgMem )
```

Draw a monochrome bitmap from a memory array.

- Draw from the bitmap buffer using the foreground color defined in the header (unset bits are transparent)

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nDstX</i>	Destination X coord for copy
in	<i>nDstY</i>	Destination Y coord for copy
in	<i>pBitmap</i>	Pointer to bitmap buffer
in	<i>bProgMem</i>	Bitmap is stored in Flash if true, RAM otherwise

## Returns

none

## 9.57.3.16 gslc\_DrvDrawPoint()

```
bool gslc_DrvDrawPoint (
    gslc_tsGui * pGui,
    int16_t nX,
    int16_t nY,
    gslc_tsColor nCol )
```

Draw a point.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nX</i>	X coordinate of point
in	<i>nY</i>	Y coordinate of point
in	<i>nCol</i>	Color RGB value to draw

## Returns

true if success, false if error

9.57.3.17 `gslc_DrvDrawPoints()`

```
bool gslc_DrvDrawPoints (
    gslc_tsGui * pGui,
    gslc_tsPt * asPt,
    uint16_t nNumPt,
    gslc_tsColor nCol )
```

Draw a point.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>asPt</i>	Array of points to draw
in	<i>n</i> ↔ <i>NumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

## Returns

true if success, false if error

9.57.3.18 `gslc_DrvDrawTxt()`

```
bool gslc_DrvDrawTxt (
    gslc_tsGui * pGui,
    int16_t nTxtX,
    int16_t nTxtY,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    gslc_tsColor colTxt,
    gslc_tsColor colBg )
```

Draw a text string at the given coordinate.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nTxtX</i>	X coordinate of top-left text string
in	<i>nTxtY</i>	Y coordinate of top-left text string
in	<i>pFont</i>	Ptr to Font
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
in	<i>colTxt</i>	Color to draw text
in	<i>colBg</i>	unused in ADAGFX, defaults to black



**Returns**

true if success, false if failure

**9.57.3.19 gslc\_DrvFontAdd()**

```
const void* gslc_DrvFontAdd (
    gslc_teFontRefType eFontRefType,
    const void * pvFontRef,
    uint16_t nFontSz )
```

Load a font from a resource and return pointer to it.

**Parameters**

in	<i>eFontRefType</i>	Font reference type (GSLC_FONTREF_PTR for Arduino)
in	<i>pvFontRef</i>	Font reference pointer (Pointer to the GFXFont array)
in	<i>nFontSz</i>	Typeface size to use

**Returns**

Void ptr to driver-specific font if load was successful, NULL otherwise

**9.57.3.20 gslc\_DrvFontsDestruct()**

```
void gslc_DrvFontsDestruct (
    gslc_tsGui * pGui )
```

Release all fonts defined in the GUI.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

none

**9.57.3.21 gslc\_DrvGetDriverDisp()**

```
void* gslc_DrvGetDriverDisp (
    gslc_tsGui * pGui )
```

Get the native display driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the display driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.57.3.22 gslc\_DrvGetDriverTouch()**

```
void* gslc_DrvGetDriverTouch (
    gslc_tsGui * pGui )
```

Get the native touch driver instance.

- This can be useful to access special commands available in the selected driver.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

Void pointer to the touch driver instance. This pointer should be typecast to the particular driver being used. If no driver was created then this function will return NULL.

**9.57.3.23 gslc\_DrvGetNameDisp()**

```
const char* gslc_DrvGetNameDisp (
    gslc_tsGui * pGui )
```

Get the display driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.57.3.24 gslc\_DrvGetNameTouch()**

```
const char* gslc_DrvGetNameTouch (
    gslc_tsGui * pGui )
```

Get the touch driver name.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

**Returns**

String containing driver name

**9.57.3.25 gslc\_DrvGetTouch()**

```
bool gslc_DrvGetTouch (
    gslc_tsGui * pGui,
    int16_t * pnX,
    int16_t * pnY,
    uint16_t * pnPress,
    gslc_teInputRawEvent * peInputEvent,
    int16_t * pnInputVal )
```

Get the last touch event from the internal touch handler.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
out	<i>pnX</i>	Ptr to X coordinate of last touch event
out	<i>pnY</i>	Ptr to Y coordinate of last touch event
out	<i>pnPress</i>	Ptr to Pressure level of last touch event (0 for none, 1 for touch)
out	<i>peInputEvent</i>	Indication of event type
out	<i>pnInputVal</i>	Additional data for event type

**Returns**

true if an event was detected or false otherwise

### 9.57.3.26 gslc\_DrvGetTxtSize()

```
bool gslc_DrvGetTxtSize (
    gslc_tsGui * pGui,
    gslc_tsFont * pFont,
    const char * pStr,
    gslc_teTxtFlags eTxtFlags,
    int16_t * pnTxtX,
    int16_t * pnTxtY,
    uint16_t * pnTxtSzW,
    uint16_t * pnTxtSzH )
```

Get the extent (width and height) of a text string.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pFont</i>	Ptr to Font structure
in	<i>pStr</i>	String to display
in	<i>eTxtFlags</i>	Flags associated with text string
out	<i>pnTxtX</i>	Ptr to offset X of text
out	<i>pnTxtY</i>	Ptr to offset Y of text
out	<i>pnTxtSzW</i>	Ptr to width of text
out	<i>pnTxtSzH</i>	Ptr to height of text

#### Returns

true if success, false if failure

### 9.57.3.27 gslc\_DrvImageDestruct()

```
void gslc_DrvImageDestruct (
    void * pvImg )
```

Release an image surface.

#### Parameters

in	<i>pvImg</i>	Void ptr to image
----	--------------	-------------------

#### Returns

none

### 9.57.3.28 gslc\_DrvInit()

```
bool gslc_DrvInit (
    gslc_tsGui * pGui )
```

Initialize the SDL library.

- Performs clean startup workaround (if enabled)
- Configures video mode
- Initializes font support

PRE:

- The environment variables should be configured before calling `gslc_DrvInit()`. This can be done with `gslc_DrvInitEnv()` or manually in user function.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

#### Returns

true if success, false if fail

### 9.57.3.29 gslc\_DrvInitTouch()

```
bool gslc_DrvInitTouch (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

#### Returns

true if successful

**9.57.3.30 gslc\_DrvInitTs()**

```
bool gslc_DrvInitTs (
    gslc_tsGui * pGui,
    const char * acDev )
```

Perform any touchscreen-specific initialization.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>acDev</i>	Device path to touchscreen eg. "/dev/input/touchscreen"

**Returns**

true if successful

**9.57.3.31 gslc\_DrvLoadImage()**

```
void* gslc_DrvLoadImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Load a bitmap (\*.bmp) and create a new image resource.

Transparency is enabled by GSLC\_BMP\_TRANS\_EN through use of color (GSLC\_BMP\_TRANS\_RGB).

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

Image pointer (surface/texture) or NULL if error

**9.57.3.32 gslc\_DrvPageFlipNow()**

```
void gslc_DrvPageFlipNow (
    gslc_tsGui * pGui )
```

Force a page flip to occur.

This generally copies active screen surface to the display.

## Parameters

in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

## Returns

none

## 9.57.3.33 gslc\_DrvRotate()

```
bool gslc_DrvRotate (
    gslc_tsGui * pGui,
    uint8_t nRotation )
```

Change rotation, automatically adapt touchscreen axes swap/flip.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nRotation</i>	Screen Rotation value (0, 1, 2 or 3)

## Returns

true if successful

## 9.57.3.34 gslc\_DrvSetBkgndColor()

```
bool gslc_DrvSetBkgndColor (
    gslc_tsGui * pGui,
    gslc_tsColor nCol )
```

Configure the background to use a solid color.

- The background is used when redrawing the entire page

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>nCol</i>	RGB Color to use

**Returns**

true if success, false if fail

**9.57.3.35 gslc\_DrvSetBkgndImage()**

```
bool gslc_DrvSetBkgndImage (
    gslc_tsGui * pGui,
    gslc_tsImgRef sImgRef )
```

Configure the background to use a bitmap image.

- The background is used when redrawing the entire page

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>sImgRef</i>	Image reference

**Returns**

true if success, false if fail

**9.57.3.36 gslc\_DrvSetClipRect()**

```
bool gslc_DrvSetClipRect (
    gslc_tsGui * pGui,
    gslc_tsRect * pRect )
```

Set the clipping rectangle for future drawing updates.

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>pRect</i>	Rectangular region to constrain edits

**Returns**

true if success, false if error



## 9.57.3.37 gslc\_DrvSetElemImageGlow()

```
bool gslc_DrvSetElemImageGlow (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's glow-state image.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

## Returns

true if success, false if error

## 9.57.3.38 gslc\_DrvSetElemImageNorm()

```
bool gslc_DrvSetElemImageNorm (
    gslc_tsGui * pGui,
    gslc_tsElem * pElem,
    gslc_tsImgRef sImgRef )
```

Set an element's normal-state image.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pElem</i>	Pointer to Element to update
in	<i>sImgRef</i>	Image reference

## Returns

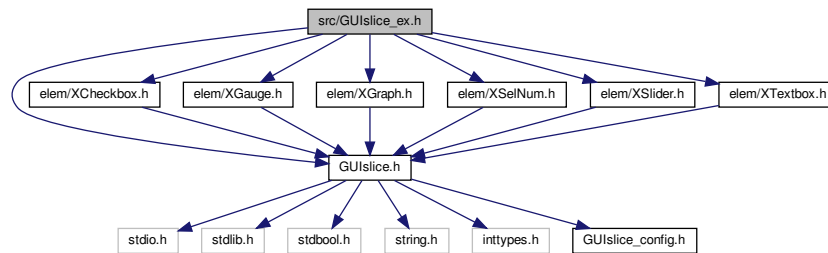
true if success, false if error

## 9.58 src/GUISlice\_ex.h File Reference

```
#include "GUISlice.h"
#include "elem/XCheckbox.h"
#include "elem/XGauge.h"
#include "elem/XGraph.h"
#include "elem/XSelNum.h"
#include "elem/XSlider.h"
```

```
#include "elem/XTextbox.h"
```

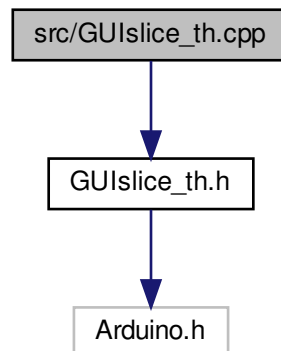
Include dependency graph for GUIslice\_ex.h:



## 9.59 src/GUIslice\_th.cpp File Reference

```
#include "GUIslice_th.h"
```

Include dependency graph for GUIslice\_th.cpp:



### Functions

- void [gslc\\_InitTouchHandler](#) ([TouchHandler](#) \*pTH)
- [TouchHandler](#) \* [gslc\\_getTouchHandler](#) (void)

### Variables

- [TouchHandler](#) \* [pTouchHandler](#)

#### 9.59.1 Function Documentation

### 9.59.1.1 gslc\_getTouchHandler()

```
TouchHandler* gslc_getTouchHandler (
    void )
```

### 9.59.1.2 gslc\_InitTouchHandler()

```
void gslc_InitTouchHandler (
    TouchHandler * pTH )
```

## 9.59.2 Variable Documentation

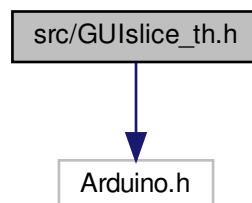
### 9.59.2.1 pTouchHandler

```
TouchHandler* pTouchHandler
```

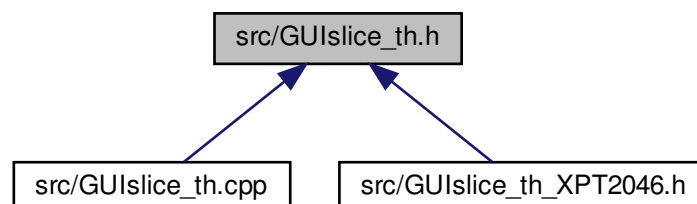
## 9.60 src/GUIslice\_th.h File Reference

```
#include <Arduino.h>
```

Include dependency graph for GUIslice\_th.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [THPoint](#)
- class [TouchHandler](#)

## Functions

- void [gslc\\_InitTouchHandler](#) ([TouchHandler](#) \*pTHO)
- [TouchHandler](#) \* [gslc\\_getTouchHandler](#) (void)

### 9.60.1 Function Documentation

#### 9.60.1.1 [gslc\\_getTouchHandler\(\)](#)

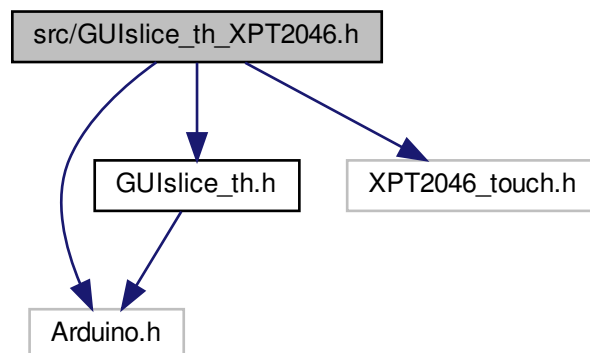
```
TouchHandler* gslc_getTouchHandler (  
    void )
```

#### 9.60.1.2 [gslc\\_InitTouchHandler\(\)](#)

```
void gslc_InitTouchHandler (  
    TouchHandler * pTHO )
```

### 9.61 [src/GUIslice\\_th\\_XPT2046.h](#) File Reference

```
#include <Arduino.h>  
#include <GUIslice_th.h>  
#include <XPT2046_touch.h>  
Include dependency graph for GUIslice\_th\_XPT2046.h:
```

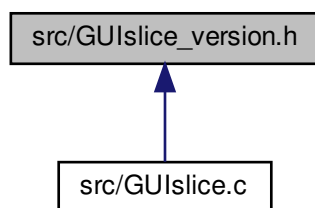


## Data Structures

- class [TouchHandler\\_XPT2046](#)

## 9.62 src/GUISlice\_version.h File Reference

This graph shows which files directly or indirectly include this file:



## Macros

- `#define` [GUISLICE\\_VER](#)

### 9.62.1 Macro Definition Documentation

#### 9.62.1.1 GUISLICE\_VER

```
#define GUISLICE_VER
```

