

GUIslice

0.13.0

Generated by Doxygen 1.8.11



# 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(const char *pFmt,...) . . . . .	14
7.1.2.2	gslc_GetDriverDisp(gslc_tsGui *pGui) . . . . .	14
7.1.2.3	gslc_GetDriverTouch(gslc_tsGui *pGui) . . . . .	14
7.1.2.4	gslc_GetNameDisp(gslc_tsGui *pGui) . . . . .	15
7.1.2.5	gslc_GetNameTouch(gslc_tsGui *pGui) . . . . .	15
7.1.2.6	gslc_GetVer(gslc_tsGui *pGui) . . . . .	15

7.1.2.7	<code>gslc_GuiRotate(gslc_tsGui *pGui, uint8_t nRotation)</code>	16
7.1.2.8	<code>gslc_Init(gslc_tsGui *pGui, void *pvDriver, gslc_tsPage *asPage, uint8_t nMax↵ Page, gslc_tsFont *asFont, uint8_t nMaxFont)</code>	16
7.1.2.9	<code>gslc_InitDebug(GSLC_CB_DEBUG_OUT pfunc)</code>	16
7.1.2.10	<code>gslc_Quit(gslc_tsGui *pGui)</code>	17
7.1.2.11	<code>gslc_SetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	17
7.1.2.12	<code>gslc_SetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	18
7.1.2.13	<code>gslc_SetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	18
7.1.2.14	<code>gslc_SetTransparentColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	18
7.1.2.15	<code>gslc_Update(gslc_tsGui *pGui)</code>	19
7.2	Graphics General Functions	20
7.2.1	Detailed Description	21
7.2.2	Function Documentation	21
7.2.2.1	<code>gslc_ClipLine(gslc_tsRect *pClipRect, int16_t *pnX0, int16_t *pnY0, int16_t↵ t *pnX1, int16_t *pnY1)</code>	21
7.2.2.2	<code>gslc_ClipPt(gslc_tsRect *pClipRect, int16_t nX, int16_t nY)</code>	21
7.2.2.3	<code>gslc_ClipRect(gslc_tsRect *pClipRect, gslc_tsRect *pRect)</code>	21
7.2.2.4	<code>gslc_ColorBlend2(gslc_tsColor colStart, gslc_tsColor colEnd, uint16_t nMidAmt, uint16_t nBlendAmt)</code>	22
7.2.2.5	<code>gslc_ColorBlend3(gslc_tsColor colStart, gslc_tsColor colMid, gslc_tsColor col↵ End, uint16_t nMidAmt, uint16_t nBlendAmt)</code>	22
7.2.2.6	<code>gslc_ColorEqual(gslc_tsColor a, gslc_tsColor b)</code>	23
7.2.2.7	<code>gslc_cosFX(int16_t n64Ang)</code>	23
7.2.2.8	<code>gslc_ExpandRect(gslc_tsRect rRect, int16_t nExpandW, int16_t nExpandH)</code>	23
7.2.2.9	<code>gslc_GetImageFromFile(const char *pFname, gslc_telmgRefFlags eFmt)</code>	24
7.2.2.10	<code>gslc_GetImageFromProg(const unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)</code>	24
7.2.2.11	<code>gslc_GetImageFromRam(unsigned char *plmgBuf, gslc_telmgRefFlags eFmt)</code>	24
7.2.2.12	<code>gslc_GetImageFromSD(const char *pFname, gslc_telmgRefFlags eFmt)</code>	24
7.2.2.13	<code>gslc_InvalidateRgnAdd(gslc_tsGui *pGui, gslc_tsRect rAddRect)</code>	25
7.2.2.14	<code>gslc_InvalidateRgnPage(gslc_tsGui *pGui, gslc_tsPage *pPage)</code>	25
7.2.2.15	<code>gslc_InvalidateRgnReset(gslc_tsGui *pGui)</code>	25
7.2.2.16	<code>gslc_InvalidateRgnScreen(gslc_tsGui *pGui)</code>	26

7.2.2.17	<a href="#">gslc_IsInRect(int16_t nSelX, int16_t nSelY, gslc_tsRect rRect)</a>	26
7.2.2.18	<a href="#">gslc_IsInWH(int16_t nSelX, int16_t nSelY, uint16_t nWidth, uint16_t nHeight)</a>	26
7.2.2.19	<a href="#">gslc_PolarToXY(uint16_t nRad, int16_t n64Ang, int16_t *nDX, int16_t *nDY)</a>	27
7.2.2.20	<a href="#">gslc_sinFX(int16_t n64Ang)</a>	27
7.2.2.21	<a href="#">gslc_UnionRect(gslc_tsRect *pRect, gslc_tsRect rAddRect)</a>	27
7.3	<a href="#">Graphics Primitive Functions</a>	29
7.3.1	<a href="#">Detailed Description</a>	30
7.3.2	<a href="#">Function Documentation</a>	30
7.3.2.1	<a href="#">gslc_DrawFillCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</a>	30
7.3.2.2	<a href="#">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)</a>	30
7.3.2.3	<a href="#">gslc_DrawFillQuad(gslc_tsGui *pGui, gslc_tsPt *psPt, gslc_tsColor nCol)</a>	31
7.3.2.4	<a href="#">gslc_DrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</a>	31
7.3.2.5	<a href="#">gslc_DrawFillRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</a>	31
7.3.2.6	<a href="#">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)</a>	32
7.3.2.7	<a href="#">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)</a>	32
7.3.2.8	<a href="#">gslc_DrawFrameCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</a>	33
7.3.2.9	<a href="#">gslc_DrawFrameQuad(gslc_tsGui *pGui, gslc_tsPt *psPt, gslc_tsColor nCol)</a>	33
7.3.2.10	<a href="#">gslc_DrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</a>	33
7.3.2.11	<a href="#">gslc_DrawFrameRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</a>	34
7.3.2.12	<a href="#">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)</a>	34
7.3.2.13	<a href="#">gslc_DrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</a>	35
7.3.2.14	<a href="#">gslc_DrawLineH(gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nW, gslc_tsColor nCol)</a>	35
7.3.2.15	<a href="#">gslc_DrawLinePolar(gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nRadStart, uint16_t nRadEnd, int16_t n64Ang, gslc_tsColor nCol)</a>	35

7.3.2.16	<code>gslc_DrawLineV(gslc_tsGui *pGui, int16_t nX, int16_t nY, uint16_t nH, gslc_tsColor nCol)</code>	36
7.3.2.17	<code>gslc_DrawSetPixel(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code>	36
7.4	Font Functions	38
7.4.1	Detailed Description	38
7.4.2	Function Documentation	38
7.4.2.1	<code>gslc_FontAdd(gslc_tsGui *pGui, int16_t nFontId, gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code>	38
7.4.2.2	<code>gslc_FontGet(gslc_tsGui *pGui, int16_t nFontId)</code>	39
7.4.2.3	<code>gslc_FontSet(gslc_tsGui *pGui, int16_t nFontId, gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code>	39
7.4.2.4	<code>gslc_FontSetMode(gslc_tsGui *pGui, int16_t nFontId, gslc_teFontRefMode eFontMode)</code>	39
7.5	Page Functions	41
7.5.1	Detailed Description	41
7.5.2	Function Documentation	41
7.5.2.1	<code>gslc_GetPageCur(gslc_tsGui *pGui)</code>	41
7.5.2.2	<code>gslc_PageAdd(gslc_tsGui *pGui, int16_t nPageId, gslc_tsElem *psElem, uint16_t nMaxElem, gslc_tsElemRef *psElemRef, uint16_t nMaxElemRef)</code>	42
7.5.2.3	<code>gslc_PageFindElemById(gslc_tsGui *pGui, int16_t nPageId, int16_t nElemId)</code>	42
7.5.2.4	<code>gslc_PageRedrawGet(gslc_tsGui *pGui)</code>	43
7.5.2.5	<code>gslc_PageRedrawSet(gslc_tsGui *pGui, bool bRedraw)</code>	43
7.5.2.6	<code>gslc_PopupHide(gslc_tsGui *pGui)</code>	43
7.5.2.7	<code>gslc_PopupShow(gslc_tsGui *pGui, int16_t nPageId, bool bModal)</code>	43
7.5.2.8	<code>gslc_SetPageBase(gslc_tsGui *pGui, int16_t nPageId)</code>	44
7.5.2.9	<code>gslc_SetPageCur(gslc_tsGui *pGui, int16_t nPageId)</code>	44
7.5.2.10	<code>gslc_SetPageOverlay(gslc_tsGui *pGui, int16_t nPageId)</code>	44
7.5.2.11	<code>gslc_SetStackPage(gslc_tsGui *pGui, uint8_t nStackPos, int16_t nPageId)</code>	45
7.5.2.12	<code>gslc_SetStackState(gslc_tsGui *pGui, uint8_t nStackPos, bool bActive, bool bDoDraw)</code>	45
7.6	Element Functions	46
7.6.1	Detailed Description	46
7.7	Element: Creation Functions	47

7.7.1	Detailed Description	47
7.7.2	Function Documentation	47
7.7.2.1	gslc_ElemCreateBox(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_↵ tsRect rElem)	47
7.7.2.2	gslc_ElemCreateBtnImg(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsRect rElem, gslc_tsImgRef slmgRef, gslc_tsImgRef slmgRefSel, GSLC_↵ _CB_TOUCH cbTouch)	48
7.7.2.3	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)	48
7.7.2.4	gslc_ElemCreateImg(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_↵ tsRect rElem, gslc_tsImgRef slmgRef)	49
7.7.2.5	gslc_ElemCreateLine(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1)	49
7.7.2.6	gslc_ElemCreateTxt(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_ts_↵ Rect rElem, char *pStrBuf, uint8_t nStrBufMax, int16_t nFontId)	50
7.8	Element: General Functions	51
7.8.1	Detailed Description	51
7.8.2	Function Documentation	51
7.8.2.1	gslc_ElemGetId(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	51
7.9	Element: Update Functions	52
7.9.1	Detailed Description	53
7.9.2	Function Documentation	53
7.9.2.1	gslc_ElemGetGlow(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	53
7.9.2.2	gslc_ElemGetGlowEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	54
7.9.2.3	gslc_ElemGetGroup(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	54
7.9.2.4	gslc_ElemGetOnScreen(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	54
7.9.2.5	gslc_ElemGetRedraw(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	55
7.9.2.6	gslc_ElemGetTxtStr(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	55
7.9.2.7	gslc_ElemGetVisible(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	55
7.9.2.8	gslc_ElemOwnsCoord(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nX, int16_t nY, bool bOnlyClickEn)	56
7.9.2.9	gslc_ElemSetClickEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool b_↵ ClickEn)	56

7.9.2.10	<code>gslc_ElemSetCol(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colFrame, gslc_tsColor colFill, gslc_tsColor colFillGlow)</code>	56
7.9.2.11	<code>gslc_ElemSetDrawFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_DRAW funcCb)</code>	57
7.9.2.12	<code>gslc_ElemSetFillEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFillEn)</code>	57
7.9.2.13	<code>gslc_ElemSetFrameEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFrameEn)</code>	58
7.9.2.14	<code>gslc_ElemSetGlow(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bGlowing)</code>	58
7.9.2.15	<code>gslc_ElemSetGlowCol(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colFrameGlow, gslc_tsColor colFillGlow, gslc_tsColor colTxtGlow)</code>	58
7.9.2.16	<code>gslc_ElemSetGlowEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bGlowEn)</code>	59
7.9.2.17	<code>gslc_ElemSetGroup(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int nGroupId)</code>	59
7.9.2.18	<code>gslc_ElemSetRedraw(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tRedrawType eRedraw)</code>	59
7.9.2.19	<code>gslc_ElemSetRoundEn(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bRoundEn)</code>	60
7.9.2.20	<code>gslc_ElemSetStyleFrom(gslc_tsGui *pGui, gslc_tsElemRef *pElemRefSrc, gslc_tsElemRef *pElemRefDest)</code>	60
7.9.2.21	<code>gslc_ElemSetTickFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_TICK funcCb)</code>	60
7.9.2.22	<code>gslc_ElemSetTouchFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_TOUCH funcCb)</code>	61
7.9.2.23	<code>gslc_ElemSetTxtAlign(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, unsigned nAlign)</code>	61
7.9.2.24	<code>gslc_ElemSetTxtCol(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colVal)</code>	62
7.9.2.25	<code>gslc_ElemSetTxtEnc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tTxtFlags eFlags)</code>	63
7.9.2.26	<code>gslc_ElemSetTxtMargin(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, unsigned nMargin)</code>	63
7.9.2.27	<code>gslc_ElemSetTxtMarginXY(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nMarginX, int8_t nMarginY)</code>	63
7.9.2.28	<code>gslc_ElemSetTxtMem(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tTxtFlags eFlags)</code>	64
7.9.2.29	<code>gslc_ElemSetTxtStr(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, const char *pStr)</code>	64
7.9.2.30	<code>gslc_ElemSetVisible(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bVisible)</code>	64



7.9.2.31	<code>gslc_ElemUpdateFont(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int nFontId)</code>	65
7.10	Touchscreen Functions	66
7.10.1	Detailed Description	66
7.10.2	Macro Definition Documentation	66
7.10.2.1	<code>TOUCH_ROTATION_DATA</code>	66
7.10.2.2	<code>TOUCH_ROTATION_DATA</code>	67
7.10.2.3	<code>TOUCH_ROTATION_FLIPX</code>	67
7.10.2.4	<code>TOUCH_ROTATION_FLIPX</code>	67
7.10.2.5	<code>TOUCH_ROTATION_FLIPY</code>	67
7.10.2.6	<code>TOUCH_ROTATION_FLIPY</code>	67
7.10.2.7	<code>TOUCH_ROTATION_SWAPXY</code>	67
7.10.2.8	<code>TOUCH_ROTATION_SWAPXY</code>	67
7.10.3	Function Documentation	67
7.10.3.1	<code>gslc_GetTouch(gslc_tsGui *pGui, int16_t *pnX, int16_t *pnY, uint16_t *pnPress, gslc_teInputRawEvent *peInputEvent, int16_t *pnInputVal)</code>	67
7.10.3.2	<code>gslc_InitTouch(gslc_tsGui *pGui, const char *acDev)</code>	67
7.10.3.3	<code>gslc_SetTouchRemapCal(gslc_tsGui *pGui, uint16_t nXMin, uint16_t nXMax, uint16_t nYMin, uint16_t nYMax)</code>	68
7.10.3.4	<code>gslc_SetTouchRemapEn(gslc_tsGui *pGui, bool bEn)</code>	68
7.10.3.5	<code>gslc_SetTouchRemapYX(gslc_tsGui *pGui, bool bSwap)</code>	68
7.11	Input Mapping Functions	70
7.11.1	Detailed Description	70
7.11.2	Function Documentation	70
7.11.2.1	<code>gslc_InitInputMap(gslc_tsGui *pGui, gslc_tsInputMap *asInputMap, uint8_t n↔InputMapMax)</code>	70
7.11.2.2	<code>gslc_InputMapAdd(gslc_tsGui *pGui, gslc_teInputRawEvent eInputEvent, int16_t nInputVal, gslc_teAction eAction, int16_t nActionVal)</code>	70
7.11.2.3	<code>gslc_SetPinPollFunc(gslc_tsGui *pGui, GSLC_CB_PIN_POLL pfunc)</code>	70
7.12	General Purpose Macros	71
7.12.1	Detailed Description	71
7.12.2	Macro Definition Documentation	71
7.12.2.1	<code>GSLC_DEBUG2_PRINT</code>	71

7.12.2.2	GSLC_DEBUG2_PRINT_CONST . . . . .	71
7.12.2.3	GSLC_DEBUG_PRINT . . . . .	71
7.12.2.4	GSLC_DEBUG_PRINT_CONST . . . . .	71
7.13	Flash-based Element Macros . . . . .	72
7.13.1	Detailed Description . . . . .	72
7.13.2	Macro Definition Documentation . . . . .	72
7.13.2.1	gslc_ElemCreateBox_P . . . . .	72
7.13.2.2	gslc_ElemCreateBtnTxt_P . . . . .	73
7.13.2.3	gslc_ElemCreateLine_P . . . . .	73
7.13.2.4	gslc_ElemCreateTxt_P . . . . .	73
7.13.2.5	gslc_ElemCreateTxt_P_R . . . . .	74
7.14	Internal Functions . . . . .	75
7.14.1	Detailed Description . . . . .	80
7.14.2	Variable Documentation . . . . .	80
7.14.2.1	abPageStackActive . . . . .	80
7.14.2.2	abPageStackDoDraw . . . . .	80
7.14.2.3	apPageStack . . . . .	80
7.14.2.4	asElem . . . . .	81
7.14.2.5	asElemRef . . . . .	81
7.14.2.6	asFont . . . . .	81
7.14.2.7	asInputMap . . . . .	81
7.14.2.8	asPage . . . . .	81
7.14.2.9	b . . . . .	81
7.14.2.10	bInvalidateEn . . . . .	81
7.14.2.11	bRedrawPartialEn . . . . .	81
7.14.2.12	bScreenNeedFlip . . . . .	81
7.14.2.13	bScreenNeedRedraw . . . . .	81
7.14.2.14	bTouchRemapEn . . . . .	82
7.14.2.15	bTouchRemapYX . . . . .	82
7.14.2.16	colElemFill . . . . .	82

7.14.2.17 colElemFillGlow . . . . .	82
7.14.2.18 colElemFrame . . . . .	82
7.14.2.19 colElemFrameGlow . . . . .	82
7.14.2.20 colElemText . . . . .	82
7.14.2.21 colElemTextGlow . . . . .	82
7.14.2.22 eAction . . . . .	82
7.14.2.23 eElemFlags . . . . .	82
7.14.2.24 eEvent . . . . .	83
7.14.2.25 eFontRefMode . . . . .	83
7.14.2.26 eFontRefType . . . . .	83
7.14.2.27 elmgFlags . . . . .	83
7.14.2.28 eInitStatTouch . . . . .	83
7.14.2.29 eTouch . . . . .	83
7.14.2.30 eTxtAlign . . . . .	83
7.14.2.31 eTxtFlags . . . . .	83
7.14.2.32 eType . . . . .	83
7.14.2.33 g . . . . .	83
7.14.2.34 h . . . . .	84
7.14.2.35 nActionVal . . . . .	84
7.14.2.36 nDisp0H . . . . .	84
7.14.2.37 nDisp0W . . . . .	84
7.14.2.38 nDispDepth . . . . .	84
7.14.2.39 nDispH . . . . .	84
7.14.2.40 nDispW . . . . .	84
7.14.2.41 nElemAutoldNext . . . . .	84
7.14.2.42 nElemCnt . . . . .	84
7.14.2.43 nElemIndFocused . . . . .	84
7.14.2.44 nElemMax . . . . .	85
7.14.2.45 nElemRefCnt . . . . .	85
7.14.2.46 nElemRefMax . . . . .	85

7.14.2.47 nFeatures . . . . .	85
7.14.2.48 nFlipX . . . . .	85
7.14.2.49 nFlipY . . . . .	85
7.14.2.50 nFontCnt . . . . .	85
7.14.2.51 nFontMax . . . . .	85
7.14.2.52 nFrameRateCnt . . . . .	85
7.14.2.53 nFrameRateStart . . . . .	85
7.14.2.54 nGroup . . . . .	86
7.14.2.55 nId . . . . .	86
7.14.2.56 nId . . . . .	86
7.14.2.57 nInputMapCnt . . . . .	86
7.14.2.58 nInputMapMax . . . . .	86
7.14.2.59 nPageCnt . . . . .	86
7.14.2.60 nPageId . . . . .	86
7.14.2.61 nPageMax . . . . .	86
7.14.2.62 nRotation . . . . .	86
7.14.2.63 nRoundRadius . . . . .	86
7.14.2.64 nSize . . . . .	87
7.14.2.65 nStrBufMax . . . . .	87
7.14.2.66 nSubType . . . . .	87
7.14.2.67 nSwapXY . . . . .	87
7.14.2.68 nTouchCalXMax . . . . .	87
7.14.2.69 nTouchCalXMin . . . . .	87
7.14.2.70 nTouchCalYMax . . . . .	87
7.14.2.71 nTouchCalYMin . . . . .	87
7.14.2.72 nTouchLastPress . . . . .	87
7.14.2.73 nTouchLastX . . . . .	87
7.14.2.74 nTouchLastY . . . . .	88
7.14.2.75 nTouchRotation . . . . .	88
7.14.2.76 nTxtMarginX . . . . .	88

7.14.2.77 nTxtMarginY . . . . .	88
7.14.2.78 nType . . . . .	88
7.14.2.79 nVal . . . . .	88
7.14.2.80 nX . . . . .	88
7.14.2.81 nY . . . . .	88
7.14.2.82 pElem . . . . .	88
7.14.2.83 pElemRefParent . . . . .	88
7.14.2.84 pElemRefTracked . . . . .	89
7.14.2.85 pFname . . . . .	89
7.14.2.86 pfuncPinPoll . . . . .	89
7.14.2.87 pfuncXDraw . . . . .	89
7.14.2.88 pfuncXEvent . . . . .	89
7.14.2.89 pfuncXTick . . . . .	89
7.14.2.90 pfuncXTouch . . . . .	89
7.14.2.91 pImgBuf . . . . .	89
7.14.2.92 pStrBuf . . . . .	89
7.14.2.93 pTxtFont . . . . .	89
7.14.2.94 pvData . . . . .	90
7.14.2.95 pvDriver . . . . .	90
7.14.2.96 pvFont . . . . .	90
7.14.2.97 pvImgRaw . . . . .	90
7.14.2.98 pvScope . . . . .	90
7.14.2.99 pXData . . . . .	90
7.14.2.100 . . . . .	90
7.14.2.101rBounds . . . . .	90
7.14.2.102Elem . . . . .	90
7.14.2.103InvalidateRect . . . . .	90
7.14.2.104sCollect . . . . .	91
7.14.2.105sElemTmpProg . . . . .	91
7.14.2.106sImgRefBkgnd . . . . .	91

7.14.2.107	<code>slmgRefGlow</code>	91
7.14.2.108	<code>slmgRefNorm</code>	91
7.14.2.109	<code>sTransCol</code>	91
7.14.2.110	<code>w</code>	91
7.14.2.111	<code>x</code>	91
7.14.2.112	<code>x</code>	91
7.14.2.113	<code>y</code>	91
7.14.2.114	<code>y</code>	91
7.15	Internal: Misc Functions	92
7.15.1	Detailed Description	92
7.15.2	Function Documentation	92
7.15.2.1	<code>gslc_ResetImage()</code>	92
7.16	Internal: Element Functions	93
7.16.1	Detailed Description	94
7.16.2	Function Documentation	94
7.16.2.1	<code>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)</code>	94
7.16.2.2	<code>gslc_ElemAdd(gslc_tsGui *pGui, int16_t nPageId, gslc_tsElem *pElem, gslc_teElemRefFlags eFlags)</code>	94
7.16.2.3	<code>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)</code>	95
7.16.2.4	<code>gslc_ElemDraw(gslc_tsGui *pGui, int16_t nPageId, int16_t nElemId)</code>	95
7.16.2.5	<code>gslc_ElemDrawByRef(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code>	96
7.16.2.6	<code>gslc_ElemSetImage(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsImgRef sImgRef, gslc_tsImgRef sImgRefSel)</code>	96
7.16.2.7	<code>gslc_GetElemFromRef(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code>	96
7.16.2.8	<code>gslc_GetElemFromRefD(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nLineNum)</code>	97
7.16.2.9	<code>gslc_GetElemRefFlag(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nFlagMask)</code>	97
7.16.2.10	<code>gslc_GetXDataFromRef(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nType, int16_t nLineNum)</code>	97

7.16.2.11	<code>gslc_SetElemRefFlag(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t n↵ FlagMask, uint8_t nFlagVal)</code>	98
7.16.2.12	<code>gslc_SetRoundRadius(gslc_tsGui *pGui, uint8_t nRadius)</code>	98
7.17	Internal: Page Functions	99
7.17.1	Detailed Description	99
7.17.2	Function Documentation	99
7.17.2.1	<code>gslc_ElemEvent(void *pvGui, gslc_tsEvent sEvent)</code>	99
7.17.2.2	<code>gslc_ElemSendEventTouch(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef↵ Tracked, gslc_teTouch eTouch, int16_t nX, int16_t nY)</code>	100
7.17.2.3	<code>gslc_EventCreate(gslc_tsGui *pGui, gslc_teEventType eType, uint8_t nSubType, void *pvScope, void *pvData)</code>	100
7.17.2.4	<code>gslc_PageEvent(void *pvGui, gslc_tsEvent sEvent)</code>	101
7.17.2.5	<code>gslc_PageFindById(gslc_tsGui *pGui, int16_t nPageId)</code>	101
7.17.2.6	<code>gslc_PageFlipGet(gslc_tsGui *pGui)</code>	101
7.17.2.7	<code>gslc_PageFlipGo(gslc_tsGui *pGui)</code>	101
7.17.2.8	<code>gslc_PageFlipSet(gslc_tsGui *pGui, bool bNeeded)</code>	102
7.17.2.9	<code>gslc_PageFocusStep(gslc_tsGui *pGui, gslc_tsPage *pPage, bool bNext)</code>	102
7.17.2.10	<code>gslc_PageRedrawCalc(gslc_tsGui *pGui)</code>	102
7.17.2.11	<code>gslc_PageRedrawGo(gslc_tsGui *pGui)</code>	103
7.18	Internal: Element Collection Functions	104
7.18.1	Detailed Description	105
7.18.2	Function Documentation	105
7.18.2.1	<code>gslc_CollectElemAdd(gslc_tsGui *pGui, gslc_tsCollect *pCollect, const gslc_ts↵ Elem *pElem, gslc_teElemRefFlags eFlags)</code>	105
7.18.2.2	<code>gslc_CollectFindElemById(gslc_tsGui *pGui, gslc_tsCollect *pCollect, int16_t↵ nElemId)</code>	105
7.18.2.3	<code>gslc_CollectFindElemFromCoord(gslc_tsGui *pGui, gslc_tsCollect *pCollect, int16_t nX, int16_t nY)</code>	105
7.18.2.4	<code>gslc_CollectFindFocusStep(gslc_tsGui *pGui, gslc_tsCollect *pCollect, bool b↵ Next, bool *pbWrapped, int16_t *pnElemInd)</code>	106
7.18.2.5	<code>gslc_CollectGetElemRefTracked(gslc_tsGui *pGui, gslc_tsCollect *pCollect)</code>	106
7.18.2.6	<code>gslc_CollectGetFocus(gslc_tsGui *pGui, gslc_tsCollect *pCollect)</code>	106
7.18.2.7	<code>gslc_CollectGetNextId(gslc_tsGui *pGui, gslc_tsCollect *pCollect)</code>	107

7.18.2.8	<code>gslc_CollectGetRedraw(gslc_tsGui *pGui, gslc_tsCollect *pCollect)</code>	107
7.18.2.9	<code>gslc_CollectReset(gslc_tsCollect *pCollect, gslc_tsElem *asElem, uint16_t nElemMax, gslc_tsElemRef *asElemRef, uint16_t nElemRefMax)</code>	107
7.18.2.10	<code>gslc_CollectSetElemTracked(gslc_tsGui *pGui, gslc_tsCollect *pCollect, gslc_tsElemRef *pElemRef)</code>	108
7.18.2.11	<code>gslc_CollectSetFocus(gslc_tsGui *pGui, gslc_tsCollect *pCollect, int16_t nElemInd)</code>	108
7.18.2.12	<code>gslc_CollectSetParent(gslc_tsGui *pGui, gslc_tsCollect *pCollect, gslc_tsElemRef *pElemRefParent)</code>	108
7.19	Internal: Element Collection Event Functions	110
7.19.1	Detailed Description	110
7.19.2	Function Documentation	110
7.19.2.1	<code>gslc_CollectEvent(void *pvGui, gslc_tsEvent sEvent)</code>	110
7.19.2.2	<code>gslc_CollectInput(gslc_tsGui *pGui, gslc_tsCollect *pCollect, gslc_tsEventTouch *pEventTouch)</code>	110
7.19.2.3	<code>gslc_CollectTouch(gslc_tsGui *pGui, gslc_tsCollect *pCollect, gslc_tsEventTouch *pEventTouch)</code>	111
7.19.2.4	<code>gslc_CollectTouchCompound(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY, gslc_tsCollect *pCollect)</code>	111
7.20	Internal: Tracking Functions	112
7.20.1	Detailed Description	112
7.20.2	Function Documentation	112
7.20.2.1	<code>gslc_InputMapLookup(gslc_tsGui *pGui, gslc_teInputRawEvent eInputEvent, int16_t nInputVal, gslc_teAction *peAction, int16_t *pnActionVal)</code>	112
7.20.2.2	<code>gslc_TrackInput(gslc_tsGui *pGui, gslc_tsPage *pPage, gslc_teInputRawEvent eInputEvent, int16_t nInputVal)</code>	112
7.20.2.3	<code>gslc_TrackTouch(gslc_tsGui *pGui, gslc_tsPage *pPage, int16_t nX, int16_t nY, uint16_t nPress)</code>	113
7.21	Internal: Cleanup Functions	114
7.21.1	Detailed Description	114
7.21.2	Function Documentation	114
7.21.2.1	<code>gslc_CollectDestruct(gslc_tsGui *pGui, gslc_tsCollect *pCollect)</code>	114
7.21.2.2	<code>gslc_ElemDestruct(gslc_tsElem *pElem)</code>	115
7.21.2.3	<code>gslc_GuiDestruct(gslc_tsGui *pGui)</code>	116
7.21.2.4	<code>gslc_PageDestruct(gslc_tsGui *pGui, gslc_tsPage *pPage)</code>	116
7.21.2.5	<code>gslc_ResetElem(gslc_tsElem *pElem)</code>	116
7.21.2.6	<code>gslc_ResetFont(gslc_tsFont *pFont)</code>	117



<b>8 Data Structure Documentation</b>	<b>119</b>
8.1 gslc_tsCollect Struct Reference . . . . .	119
8.1.1 Detailed Description . . . . .	120
8.2 gslc_tsColor Struct Reference . . . . .	120
8.2.1 Detailed Description . . . . .	120
8.3 gslc_tsDriver Struct Reference . . . . .	121
8.3.1 Field Documentation . . . . .	121
8.3.1.1 nColBkgnd . . . . .	121
8.3.1.2 rClipRect . . . . .	121
8.4 gslc_tsElem Struct Reference . . . . .	122
8.4.1 Detailed Description . . . . .	123
8.5 gslc_tsElemRef Struct Reference . . . . .	124
8.5.1 Detailed Description . . . . .	124
8.6 gslc_tsEvent Struct Reference . . . . .	124
8.6.1 Detailed Description . . . . .	125
8.7 gslc_tsEventTouch Struct Reference . . . . .	125
8.7.1 Detailed Description . . . . .	125
8.8 gslc_tsFont Struct Reference . . . . .	125
8.8.1 Detailed Description . . . . .	126
8.9 gslc_tsGui Struct Reference . . . . .	126
8.9.1 Detailed Description . . . . .	128
8.10 gslc_tsImgRef Struct Reference . . . . .	128
8.10.1 Detailed Description . . . . .	129
8.11 gslc_tsInputMap Struct Reference . . . . .	129
8.11.1 Detailed Description . . . . .	129
8.12 gslc_tsPage Struct Reference . . . . .	130
8.12.1 Detailed Description . . . . .	130
8.13 gslc_tsPt Struct Reference . . . . .	131
8.13.1 Detailed Description . . . . .	131
8.14 gslc_tsRect Struct Reference . . . . .	131

8.14.1 Detailed Description . . . . .	131
8.15 gslc_tsXCheckbox Struct Reference . . . . .	132
8.15.1 Detailed Description . . . . .	132
8.15.2 Field Documentation . . . . .	132
8.15.2.1 bChecked . . . . .	132
8.15.2.2 bRadio . . . . .	132
8.15.2.3 colCheck . . . . .	133
8.15.2.4 nStyle . . . . .	133
8.15.2.5 pfuncXToggle . . . . .	133
8.16 gslc_tsXGauge Struct Reference . . . . .	133
8.16.1 Detailed Description . . . . .	134
8.16.2 Field Documentation . . . . .	134
8.16.2.1 bFlip . . . . .	134
8.16.2.2 bIndicFill . . . . .	134
8.16.2.3 bValLastValid . . . . .	135
8.16.2.4 bVert . . . . .	135
8.16.2.5 colGauge . . . . .	135
8.16.2.6 colTick . . . . .	135
8.16.2.7 nIndicLen . . . . .	135
8.16.2.8 nIndicTip . . . . .	135
8.16.2.9 nMax . . . . .	135
8.16.2.10 nMin . . . . .	135
8.16.2.11 nStyle . . . . .	135
8.16.2.12 nTickCnt . . . . .	135
8.16.2.13 nTickLen . . . . .	136
8.16.2.14 nVal . . . . .	136
8.16.2.15 nValLast . . . . .	136
8.17 gslc_tsXGlowball Struct Reference . . . . .	136
8.17.1 Detailed Description . . . . .	137
8.17.2 Field Documentation . . . . .	137

8.17.2.1	colBg	137
8.17.2.2	nAngEnd	137
8.17.2.3	nAngStart	137
8.17.2.4	nMidX	137
8.17.2.5	nMidY	138
8.17.2.6	nNumRings	138
8.17.2.7	nQuality	138
8.17.2.8	nVal	138
8.17.2.9	nValLast	138
8.17.2.10	pRings	138
8.18	gslc_tsXGlowballRing Struct Reference	138
8.18.1	Field Documentation	139
8.18.1.1	cCol	139
8.18.1.2	nRad1	139
8.18.1.3	nRad2	139
8.19	gslc_tsXGraph Struct Reference	139
8.19.1	Detailed Description	140
8.19.2	Field Documentation	140
8.19.2.1	bScrollEn	140
8.19.2.2	colGraph	140
8.19.2.3	eStyle	141
8.19.2.4	nBufCnt	141
8.19.2.5	nBufMax	141
8.19.2.6	nMargin	141
8.19.2.7	nPlotIndMax	141
8.19.2.8	nPlotIndStart	141
8.19.2.9	nPlotValMax	141
8.19.2.10	nPlotValMin	141
8.19.2.11	nScrollPos	141
8.19.2.12	nWndHeight	141

8.19.2.13 nWndWidth . . . . .	142
8.19.2.14 pBuf . . . . .	142
8.20 gslc_tsXListbox Struct Reference . . . . .	142
8.20.1 Detailed Description . . . . .	143
8.20.2 Field Documentation . . . . .	143
8.20.2.1 bltemAutoSizeH . . . . .	143
8.20.2.2 bltemAutoSizeW . . . . .	143
8.20.2.3 bNeedRecalc . . . . .	143
8.20.2.4 colGap . . . . .	144
8.20.2.5 nBufItemsMax . . . . .	144
8.20.2.6 nBufItemsPos . . . . .	144
8.20.2.7 nCols . . . . .	144
8.20.2.8 nItemCnt . . . . .	144
8.20.2.9 nItemCurSel . . . . .	144
8.20.2.10 nItemCurSelLast . . . . .	144
8.20.2.11 nItemGap . . . . .	144
8.20.2.12 nItemH . . . . .	144
8.20.2.13 nItemSavedSel . . . . .	144
8.20.2.14 nItemTop . . . . .	145
8.20.2.15 nItemW . . . . .	145
8.20.2.16 nMarginH . . . . .	145
8.20.2.17 nMarginW . . . . .	145
8.20.2.18 nRows . . . . .	145
8.20.2.19 pBufItems . . . . .	145
8.20.2.20 pfuncXSel . . . . .	145
8.21 gslc_tsXProgress Struct Reference . . . . .	145
8.21.1 Detailed Description . . . . .	146
8.21.2 Field Documentation . . . . .	146
8.21.2.1 bFlip . . . . .	146
8.21.2.2 bValLastValid . . . . .	146

8.21.2.3	bVert	146
8.21.2.4	colGauge	146
8.21.2.5	nMax	146
8.21.2.6	nMin	147
8.21.2.7	nVal	147
8.21.2.8	nValLast	147
8.22	gslc_tsXRadial Struct Reference	147
8.22.1	Detailed Description	148
8.22.2	Field Documentation	148
8.22.2.1	bFlip	148
8.22.2.2	bIndicFill	148
8.22.2.3	bValLastValid	148
8.22.2.4	colGauge	149
8.22.2.5	colTick	149
8.22.2.6	nIndicLen	149
8.22.2.7	nIndicTip	149
8.22.2.8	nMax	149
8.22.2.9	nMin	149
8.22.2.10	nTickCnt	149
8.22.2.11	nTickLen	149
8.22.2.12	nVal	149
8.22.2.13	nValLast	149
8.23	gslc_tsXRamp Struct Reference	150
8.23.1	Detailed Description	150
8.23.2	Field Documentation	150
8.23.2.1	bValLastValid	150
8.23.2.2	nMax	150
8.23.2.3	nMin	150
8.23.2.4	nVal	150
8.23.2.5	nValLast	151

8.24	gslc_tsXRingGauge Struct Reference . . . . .	151
8.24.1	Detailed Description . . . . .	152
8.24.2	Field Documentation . . . . .	152
8.24.2.1	acStrLast . . . . .	152
8.24.2.2	bGradient . . . . .	152
8.24.2.3	colRing1 . . . . .	152
8.24.2.4	colRing2 . . . . .	152
8.24.2.5	colRingRemain . . . . .	152
8.24.2.6	nAngRange . . . . .	152
8.24.2.7	nAngStart . . . . .	152
8.24.2.8	nQuality . . . . .	152
8.24.2.9	nSegGap . . . . .	152
8.24.2.10	nThickness . . . . .	152
8.24.2.11	nVal . . . . .	152
8.24.2.12	nValLast . . . . .	152
8.24.2.13	nValMax . . . . .	152
8.24.2.14	nValMin . . . . .	152
8.25	gslc_tsXSlider Struct Reference . . . . .	153
8.25.1	Detailed Description . . . . .	153
8.25.2	Field Documentation . . . . .	154
8.25.2.1	bTrim . . . . .	154
8.25.2.2	bVert . . . . .	154
8.25.2.3	colTick . . . . .	154
8.25.2.4	colTrim . . . . .	154
8.25.2.5	nPos . . . . .	154
8.25.2.6	nPosMax . . . . .	154
8.25.2.7	nPosMin . . . . .	154
8.25.2.8	nThumbSz . . . . .	154
8.25.2.9	nTickDiv . . . . .	154
8.25.2.10	nTickLen . . . . .	154

8.25.2.11 pfuncXPos . . . . .	155
8.26 gslc_tsXTemplate Struct Reference . . . . .	155
8.26.1 Detailed Description . . . . .	155
8.27 gslc_tsXTextbox Struct Reference . . . . .	155
8.27.1 Detailed Description . . . . .	156
8.27.2 Field Documentation . . . . .	156
8.27.2.1 bScrollEn . . . . .	156
8.27.2.2 bWrapEn . . . . .	156
8.27.2.3 nBufCols . . . . .	156
8.27.2.4 nBufPosX . . . . .	156
8.27.2.5 nBufPosY . . . . .	157
8.27.2.6 nBufRows . . . . .	157
8.27.2.7 nChSizeX . . . . .	157
8.27.2.8 nChSizeY . . . . .	157
8.27.2.9 nCurPosX . . . . .	157
8.27.2.10 nCurPosY . . . . .	157
8.27.2.11 nMarginX . . . . .	157
8.27.2.12 nMarginY . . . . .	157
8.27.2.13 nRedrawRow . . . . .	157
8.27.2.14 nScrollPos . . . . .	157
8.27.2.15 nWndCols . . . . .	158
8.27.2.16 nWndRows . . . . .	158
8.27.2.17 nWndRowStart . . . . .	158
8.27.2.18 pBuf . . . . .	158
8.28 THPoint Class Reference . . . . .	158
8.28.1 Constructor & Destructor Documentation . . . . .	159
8.28.1.1 THPoint(void) . . . . .	159
8.28.1.2 THPoint(uint16_t x, uint16_t y, uint16_t z) . . . . .	159
8.28.2 Member Function Documentation . . . . .	159
8.28.2.1 operator!=(THPoint) . . . . .	159

8.28.2.2	<code>operator==(THPoint)</code>	159
8.28.3	Field Documentation	159
8.28.3.1	<code>x</code>	159
8.28.3.2	<code>y</code>	159
8.28.3.3	<code>z</code>	159
8.29	TouchHandler Class Reference	159
8.29.1	Constructor & Destructor Documentation	160
8.29.1.1	<code>TouchHandler()</code>	160
8.29.2	Member Function Documentation	160
8.29.2.1	<code>begin(void)</code>	160
8.29.2.2	<code>getPoint(void)</code>	160
8.29.2.3	<code>scale(THPoint pln)</code>	160
8.29.2.4	<code>setCalibration(uint16_t ts_xMin, uint16_t ts_xMax, uint16_t ts_yMin, uint16_t ts_yMax)</code>	160
8.29.2.5	<code>setSize(uint16_t _disp_xSize, uint16_t _disp_ySize)</code>	160
8.29.2.6	<code>setSwapFlip(bool _swapXY, bool _flipX, bool _flipY)</code>	160
8.30	TouchHandler_XPT2046 Class Reference	161
8.30.1	Constructor & Destructor Documentation	162
8.30.1.1	<code>TouchHandler_XPT2046(SPIClass &amp;spi, uint8_t spi_cs_pin)</code>	162
8.30.2	Member Function Documentation	162
8.30.2.1	<code>begin(void)</code>	162
8.30.2.2	<code>getPoint(void)</code>	162
8.30.3	Field Documentation	162
8.30.3.1	<code>spi</code>	162
8.30.3.2	<code>touchDriver</code>	162



<b>9</b>	<b>File Documentation</b>	<b>163</b>
9.1	README.md File Reference	163
9.2	src/elem/XCheckbox.c File Reference	163
9.2.1	Function Documentation	164
9.2.1.1	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)	164
9.2.1.2	gslc_ElemXCheckboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	164
9.2.1.3	gslc_ElemXCheckboxFindChecked(gslc_tsGui *pGui, int16_t nGroupId)	165
9.2.1.4	gslc_ElemXCheckboxGetState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	165
9.2.1.5	gslc_ElemXCheckboxSetState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bChecked)	165
9.2.1.6	gslc_ElemXCheckboxSetStateFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XCHECKBOX pfuncCb)	166
9.2.1.7	gslc_ElemXCheckboxSetStateHelp(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bChecked)	166
9.2.1.8	gslc_ElemXCheckboxToggleState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	166
9.2.1.9	gslc_ElemXCheckboxTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)	166
9.2.2	Variable Documentation	167
9.2.2.1	ERRSTR_NULL	167
9.2.2.2	ERRSTR_PXD_NULL	167
9.3	src/elem/XCheckbox.h File Reference	167
9.3.1	Macro Definition Documentation	168
9.3.1.1	gslc_ElemXCheckboxCreate_P	168
9.3.1.2	GSLC_TYPEX_CHECKBOX	169
9.3.2	Typedef Documentation	169
9.3.2.1	GSLC_CB_XCHECKBOX	169
9.3.3	Enumeration Type Documentation	169
9.3.3.1	gslc_teXCheckboxStyle	169
9.3.4	Function Documentation	170

9.3.4.1	<code>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)</code> . . . . .	170
9.3.4.2	<code>gslc_ElemXCheckboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	171
9.3.4.3	<code>gslc_ElemXCheckboxFindChecked(gslc_tsGui *pGui, int16_t nGroupId)</code> . . . . .	171
9.3.4.4	<code>gslc_ElemXCheckboxGetState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . . . . .	172
9.3.4.5	<code>gslc_ElemXCheckboxSetState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bChecked)</code> . . . . .	172
9.3.4.6	<code>gslc_ElemXCheckboxSetStateFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XCHECKBOX pfuncCb)</code> . . . . .	172
9.3.4.7	<code>gslc_ElemXCheckboxToggleState(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . . . . .	172
9.3.4.8	<code>gslc_ElemXCheckboxTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)</code> . . . . .	173
9.4	<code>src/elem/XGauge.c</code> File Reference . . . . .	173
9.4.1	Function Documentation . . . . .	175
9.4.1.1	<code>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)</code> . . . . .	175
9.4.1.2	<code>gslc_ElemXGaugeDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	175
9.4.1.3	<code>gslc_ElemXGaugeDrawProgressBar(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	176
9.4.1.4	<code>gslc_ElemXGaugeSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip)</code> . . . . .	176
9.4.1.5	<code>gslc_ElemXGaugeSetIndicator(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colGauge, uint16_t nIndicLen, uint16_t nIndicTip, bool bIndicFill)</code> . . . . .	176
9.4.1.6	<code>gslc_ElemXGaugeSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teXGaugeStyle nStyle)</code> . . . . .	177
9.4.1.7	<code>gslc_ElemXGaugeSetTicks(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colTick, uint16_t nTickCnt, uint16_t nTickLen)</code> . . . . .	177
9.4.1.8	<code>gslc_ElemXGaugeUpdate(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code> . . . . .	178
9.4.2	Variable Documentation . . . . .	178
9.4.2.1	<code>ERRSTR_NULL</code> . . . . .	178
9.4.2.2	<code>ERRSTR_PXD_NULL</code> . . . . .	178
9.5	<code>src/elem/XGauge.h</code> File Reference . . . . .	178

9.5.1	Macro Definition Documentation	180
9.5.1.1	gslc_ElemXGaugeCreate_P	180
9.5.1.2	GSLC_TYPEX_GAUGE	180
9.5.2	Enumeration Type Documentation	180
9.5.2.1	gslc_teXGaugeStyle	180
9.5.3	Function Documentation	181
9.5.3.1	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)	181
9.5.3.2	gslc_ElemXGaugeDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	181
9.5.3.3	gslc_ElemXGaugeDrawProgressBar(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)	182
9.5.3.4	gslc_ElemXGaugeSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip)	182
9.5.3.5	gslc_ElemXGaugeSetIndicator(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colGauge, uint16_t nIndicLen, uint16_t nIndicTip, bool bIndicFill)	182
9.5.3.6	gslc_ElemXGaugeSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teXGaugeStyle nType)	183
9.5.3.7	gslc_ElemXGaugeSetTicks(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colTick, uint16_t nTickCnt, uint16_t nTickLen)	183
9.5.3.8	gslc_ElemXGaugeUpdate(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)	184
9.6	src/elem/XGlowball.c File Reference	184
9.6.1	Function Documentation	185
9.6.1.1	drawXGlowball(gslc_tsGui *pGui, gslc_tsXGlowball *pGlowball, int16_t nMidX, int16_t nMidY, int16_t nVal, uint16_t nAngStart, uint16_t nAngEnd)	185
9.6.1.2	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)	185
9.6.1.3	drawXGlowballRing(gslc_tsGui *pGui, gslc_tsXGlowball *pGlowball, int16_t nMidX, int16_t nMidY, int16_t nVal, uint16_t nAngStart, uint16_t nAngEnd, bool bErase)	185
9.6.1.4	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)	185
9.6.1.5	gslc_ElemXGlowballDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	186

9.6.1.6	<code>gslc_ElemXGlowballSetAngles(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nAngStart, int16_t nAngEnd)</code>	186
9.6.1.7	<code>gslc_ElemXGlowballSetColorBack(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colBg)</code>	186
9.6.1.8	<code>gslc_ElemXGlowballSetQuality(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nQuality)</code>	186
9.6.1.9	<code>gslc_ElemXGlowballSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	186
9.6.2	Variable Documentation	186
9.6.2.1	<code>ERRSTR_NULL</code>	186
9.6.2.2	<code>ERRSTR_PXD_NULL</code>	186
9.7	<code>src/elem/XGlowball.h</code> File Reference	186
9.7.1	Macro Definition Documentation	188
9.7.1.1	<code>GSLC_TYPEX_GLOW</code>	188
9.7.2	Function Documentation	188
9.7.2.1	<code>drawXGlowball(gslc_tsGui *pGui, gslc_tsXGlowball *pGlowball, int16_t nMidX, int16_t nMidY, int16_t nVal, uint16_t nAngStart, uint16_t nAngEnd)</code>	188
9.7.2.2	<code>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)</code>	188
9.7.2.3	<code>drawXGlowballRing(gslc_tsGui *pGui, gslc_tsXGlowball *pGlowball, int16_t nMidX, int16_t nMidY, int16_t nVal, uint16_t nAngStart, uint16_t nAngEnd, bool bErase)</code>	188
9.7.2.4	<code>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)</code>	188
9.7.2.5	<code>gslc_ElemXGlowballDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code>	188
9.7.2.6	<code>gslc_ElemXGlowballSetAngles(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nAngStart, int16_t nAngEnd)</code>	189
9.7.2.7	<code>gslc_ElemXGlowballSetColorBack(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colBg)</code>	189
9.7.2.8	<code>gslc_ElemXGlowballSetQuality(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nQuality)</code>	189
9.7.2.9	<code>gslc_ElemXGlowballSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	189
9.8	<code>src/elem/XGraph.c</code> File Reference	189
9.8.1	Function Documentation	190

9.8.1.1	<a href="#">gslc_ElemXGraphAdd(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</a>	190
9.8.1.2	<a href="#">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)</a>	190
9.8.1.3	<a href="#">gslc_ElemXGraphDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↔Redraw)</a>	190
9.8.1.4	<a href="#">gslc_ElemXGraphScrollSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)</a>	192
9.8.1.5	<a href="#">gslc_ElemXGraphSetRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nYMin, int16_t nYMax)</a>	192
9.8.1.6	<a href="#">gslc_ElemXGraphSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc↔_teXGraphStyle eStyle, uint8_t nMargin)</a>	192
9.8.2	<a href="#">Variable Documentation</a>	193
9.8.2.1	<a href="#">ERRSTR_NULL</a>	193
9.8.2.2	<a href="#">ERRSTR_PXD_NULL</a>	193
9.9	<a href="#">src/elem/XGraph.h File Reference</a>	193
9.9.1	<a href="#">Macro Definition Documentation</a>	194
9.9.1.1	<a href="#">GSLC_TYPEX_GRAPH</a>	194
9.9.2	<a href="#">Enumeration Type Documentation</a>	194
9.9.2.1	<a href="#">gslc_teXGraphStyle</a>	194
9.9.3	<a href="#">Function Documentation</a>	194
9.9.3.1	<a href="#">gslc_ElemXGraphAdd(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</a>	194
9.9.3.2	<a href="#">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)</a>	195
9.9.3.3	<a href="#">gslc_ElemXGraphDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↔Redraw)</a>	195
9.9.3.4	<a href="#">gslc_ElemXGraphScrollSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)</a>	196
9.9.3.5	<a href="#">gslc_ElemXGraphSetRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nYMin, int16_t nYMax)</a>	196
9.9.3.6	<a href="#">gslc_ElemXGraphSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc↔_teXGraphStyle eStyle, uint8_t nMargin)</a>	196
9.10	<a href="#">src/elem/XKeyPad.c File Reference</a>	197
9.11	<a href="#">src/elem/XKeyPad.h File Reference</a>	197
9.12	<a href="#">src/elem/XKeyPad_Alpha.c File Reference</a>	198

9.13	src/elem/XKeyPad_Alpha.h File Reference	198
9.14	src/elem/XKeyPad_Num.c File Reference	199
9.15	src/elem/XKeyPad_Num.h File Reference	200
9.16	src/elem/XListbox.c File Reference	201
9.16.1	Macro Definition Documentation	202
9.16.1.1	XLISTBOX_MAX_STR	202
9.16.2	Function Documentation	202
9.16.2.1	gslc_ElemXListboxAddItem(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, const char *pStrItem)	202
9.16.2.2	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)	203
9.16.2.3	gslc_ElemXListboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	203
9.16.2.4	gslc_ElemXListboxGetItem(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemCurSel, char *pStrItem, uint8_t nStrItemLen)	204
9.16.2.5	gslc_ElemXListboxGetItemCnt(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	204
9.16.2.6	gslc_ElemXListboxGetSel(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	204
9.16.2.7	gslc_ElemXListboxItemsSetGap(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nGap, gslc_tsColor colGap)	205
9.16.2.8	gslc_ElemXListboxItemsSetSize(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemW, int16_t nItemH)	205
9.16.2.9	gslc_ElemXListboxRecalcSize(gslc_tsXListbox *pListbox, gslc_tsRect rElem)	205
9.16.2.10	gslc_ElemXListboxReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	205
9.16.2.11	gslc_ElemXListboxSetMargin(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nMarginW, int8_t nMarginH)	206
9.16.2.12	gslc_ElemXListboxSetScrollPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nScrollPos)	206
9.16.2.13	gslc_ElemXListboxSetSel(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemCurSel)	206
9.16.2.14	gslc_ElemXListboxSetSelFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XLISTBOX_SEL funcCb)	207
9.16.2.15	gslc_ElemXListboxSetSize(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nRows, int8_t nCols)	207
9.16.2.16	gslc_ElemXListboxTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)	207

9.16.3	Variable Documentation	208
9.16.3.1	ERRSTR_NULL	208
9.16.3.2	ERRSTR_PXD_NULL	208
9.17	src/elem/XListbox.h File Reference	208
9.17.1	Macro Definition Documentation	210
9.17.1.1	GSLC_TYPEX_LISTBOX	210
9.17.1.2	XLISTBOX_BUF_OH_R	210
9.17.1.3	XLISTBOX_SEL_NONE	210
9.17.1.4	XLISTBOX_SIZE_AUTO	210
9.17.2	Typedef Documentation	210
9.17.2.1	GSLC_CB_XLISTBOX_SEL	210
9.17.3	Function Documentation	210
9.17.3.1	gslc_ElemXListboxAddItem(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, const char *pStrItem)	210
9.17.3.2	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)	210
9.17.3.3	gslc_ElemXListboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	211
9.17.3.4	gslc_ElemXListboxGetItem(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemCurSel, char *pStrItem, uint8_t nStrItemLen)	211
9.17.3.5	gslc_ElemXListboxGetItemCnt(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	212
9.17.3.6	gslc_ElemXListboxGetSel(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	213
9.17.3.7	gslc_ElemXListboxItemsSetGap(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nGap, gslc_tsColor colGap)	213
9.17.3.8	gslc_ElemXListboxItemsSetSize(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemW, int16_t nItemH)	213
9.17.3.9	gslc_ElemXListboxReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	214
9.17.3.10	gslc_ElemXListboxSetMargin(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nMarginW, int8_t nMarginH)	214
9.17.3.11	gslc_ElemXListboxSetScrollPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nScrollPos)	214
9.17.3.12	gslc_ElemXListboxSetSel(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nItemCurSel)	215

9.17.3.13	gslc_ElemXListboxSetSelFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XLISTBOX_SEL funcCb) . . . . .	215
9.17.3.14	gslc_ElemXListboxSetSize(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nRows, int8_t nCols) . . . . .	215
9.17.3.15	gslc_ElemXListboxTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY) . . . . .	216
9.18	src/elem/XProgress.c File Reference . . . . .	216
9.18.1	Function Documentation . . . . .	217
9.18.1.1	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) . . . . .	217
9.18.1.2	gslc_ElemXProgressDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw) . . . . .	218
9.18.1.3	gslc_ElemXProgressDrawHelp(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw) . . . . .	218
9.18.1.4	gslc_ElemXProgressSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip) . . . . .	219
9.18.1.5	gslc_ElemXProgressSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal) . . . . .	219
9.18.2	Variable Documentation . . . . .	219
9.18.2.1	ERRSTR_NULL . . . . .	219
9.18.2.2	ERRSTR_PXD_NULL . . . . .	219
9.19	src/elem/XProgress.h File Reference . . . . .	220
9.19.1	Macro Definition Documentation . . . . .	221
9.19.1.1	gslc_ElemXProgressCreate_P . . . . .	221
9.19.1.2	GSLC_TYPEX_PROGRESS . . . . .	221
9.19.2	Function Documentation . . . . .	222
9.19.2.1	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) . . . . .	222
9.19.2.2	gslc_ElemXProgressDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw) . . . . .	222
9.19.2.3	gslc_ElemXProgressDrawHelp(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw) . . . . .	223
9.19.2.4	gslc_ElemXProgressSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip) . . . . .	223



9.19.2.5	<code>gslc_ElemXProgressSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	223
9.20	<code>src/elem/XRadial.c</code> File Reference	224
9.20.1	Function Documentation	225
9.20.1.1	<code>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)</code>	225
9.20.1.2	<code>gslc_ElemXRadialDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↔Redraw)</code>	225
9.20.1.3	<code>gslc_ElemXRadialDrawRadial(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code>	226
9.20.1.4	<code>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)</code>	226
9.20.1.5	<code>gslc_ElemXRadialSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip)</code>	226
9.20.1.6	<code>gslc_ElemXRadialSetIndicator(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colGauge, uint16_t nIndicLen, uint16_t nIndicTip, bool bIndicFill)</code>	227
9.20.1.7	<code>gslc_ElemXRadialSetTicks(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc↔_tsColor colTick, uint16_t nTickCnt, uint16_t nTickLen)</code>	227
9.20.1.8	<code>gslc_ElemXRadialSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	227
9.20.2	Variable Documentation	228
9.20.2.1	<code>ERRSTR_NULL</code>	228
9.20.2.2	<code>ERRSTR_PXD_NULL</code>	228
9.21	<code>src/elem/XRadial.h</code> File Reference	228
9.21.1	Macro Definition Documentation	229
9.21.1.1	<code>gslc_ElemXRadialCreate_P</code>	229
9.21.1.2	<code>GSLC_TYPEX_RADIAL</code>	230
9.21.2	Function Documentation	230
9.21.2.1	<code>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)</code>	230
9.21.2.2	<code>gslc_ElemXRadialDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↔Redraw)</code>	230
9.21.2.3	<code>gslc_ElemXRadialDrawRadial(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code>	231

9.21.2.4	<code>gslc_ElemXRadialSetFlip(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bFlip)</code>	231
9.21.2.5	<code>gslc_ElemXRadialSetIndicator(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colGauge, uint16_t nIndicLen, uint16_t nIndicTip, bool bIndicFill)</code>	232
9.21.2.6	<code>gslc_ElemXRadialSetTicks(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colTick, uint16_t nTickCnt, uint16_t nTickLen)</code>	232
9.21.2.7	<code>gslc_ElemXRadialSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	232
9.22	<code>src/elem/XRamp.c</code> File Reference	233
9.22.1	Function Documentation	234
9.22.1.1	<code>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)</code>	234
9.22.1.2	<code>gslc_ElemXRampDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code>	234
9.22.1.3	<code>gslc_ElemXRampDrawHelp(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code>	235
9.22.1.4	<code>gslc_ElemXRampSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	235
9.22.2	Variable Documentation	235
9.22.2.1	<code>ERRSTR_NULL</code>	235
9.22.2.2	<code>ERRSTR_PXD_NULL</code>	235
9.23	<code>src/elem/XRamp.h</code> File Reference	236
9.23.1	Macro Definition Documentation	237
9.23.1.1	<code>gslc_ElemXRampCreate_P</code>	237
9.23.1.2	<code>GSLC_TYPEX_RAMP</code>	237
9.23.2	Function Documentation	237
9.23.2.1	<code>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)</code>	237
9.23.2.2	<code>gslc_ElemXRampDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code>	239
9.23.2.3	<code>gslc_ElemXRampDrawHelp(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_teRedrawType eRedraw)</code>	239
9.23.2.4	<code>gslc_ElemXRampSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code>	240
9.24	<code>src/elem/XRingGauge.c</code> File Reference	240

9.24.1	Function Documentation	241
9.24.1.1	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)	241
9.24.1.2	gslc_ElemXRingGaugeDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	242
9.24.1.3	gslc_ElemXRingGaugeSetAngleRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nStart, int16_t nRange, bool bClockwise)	242
9.24.1.4	gslc_ElemXRingGaugeSetColorActiveFlat(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colActive)	243
9.24.1.5	gslc_ElemXRingGaugeSetColorActiveGradient(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colStart, gslc_tsColor colEnd)	243
9.24.1.6	gslc_ElemXRingGaugeSetColorInactive(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colInactive)	243
9.24.1.7	gslc_ElemXRingGaugeSetQuality(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nSegments)	244
9.24.1.8	gslc_ElemXRingGaugeSetThickness(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nThickness)	244
9.24.1.9	gslc_ElemXRingGaugeSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)	245
9.24.1.10	gslc_ElemXRingGaugeSetValRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nValMin, int16_t nValMax)	245
9.24.2	Variable Documentation	245
9.24.2.1	ERRSTR_NULL	245
9.24.2.2	ERRSTR_PXD_NULL	245
9.25	src/elem/XRingGauge.h File Reference	246
9.25.1	Macro Definition Documentation	247
9.25.1.1	GSLC_TYPEX_RING	247
9.25.1.2	XRING_STR_MAX	247
9.25.2	Function Documentation	247
9.25.2.1	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)	247
9.25.2.2	gslc_ElemXRingGaugeDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	248
9.25.2.3	gslc_ElemXRingGaugeSetAngleRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nStart, int16_t nRange, bool bClockwise)	248

9.25.2.4	<code>gslc_ElemXRingGaugeSetColorActiveFlat(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colActive)</code> . . . . .	249
9.25.2.5	<code>gslc_ElemXRingGaugeSetColorActiveGradient(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colStart, gslc_tsColor colEnd)</code> . . . . .	249
9.25.2.6	<code>gslc_ElemXRingGaugeSetColorInactive(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor colInactive)</code> . . . . .	249
9.25.2.7	<code>gslc_ElemXRingGaugeSetQuality(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint16_t nSegments)</code> . . . . .	250
9.25.2.8	<code>gslc_ElemXRingGaugeSetThickness(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int8_t nThickness)</code> . . . . .	250
9.25.2.9	<code>gslc_ElemXRingGaugeSetVal(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nVal)</code> . . . . .	251
9.25.2.10	<code>gslc_ElemXRingGaugeSetValRange(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nValMin, int16_t nValMax)</code> . . . . .	251
9.26	<code>src/elem/XSelNum.c</code> File Reference . . . . .	251
9.26.1	Variable Documentation . . . . .	252
9.26.1.1	<code>ERRSTR_NULL</code> . . . . .	252
9.26.1.2	<code>ERRSTR_PXD_NULL</code> . . . . .	252
9.27	<code>src/elem/XSelNum.h</code> File Reference . . . . .	252
9.28	<code>src/elem/XSlider.c</code> File Reference . . . . .	253
9.28.1	Function Documentation . . . . .	254
9.28.1.1	<code>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)</code> . . . . .	254
9.28.1.2	<code>gslc_ElemXSliderDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	254
9.28.1.3	<code>gslc_ElemXSliderGetPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . . . . .	255
9.28.1.4	<code>gslc_ElemXSliderSetPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nPos)</code> . . . . .	255
9.28.1.5	<code>gslc_ElemXSliderSetPosFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XSLIDER_POS funcCb)</code> . . . . .	255
9.28.1.6	<code>gslc_ElemXSliderSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bTrim, gslc_tsColor colTrim, uint16_t nTickDiv, int16_t nTickLen, gslc_tsColor colTick)</code> . . . . .	256
9.28.1.7	<code>gslc_ElemXSliderTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)</code> . . . . .	256
9.28.2	Variable Documentation . . . . .	257

9.28.2.1	ERRSTR_NULL	257
9.28.2.2	ERRSTR_PXD_NULL	257
9.29	src/elem/XSlider.h File Reference	257
9.29.1	Macro Definition Documentation	258
9.29.1.1	gslc_ElemXSliderCreate_P	258
9.29.1.2	GSLC_TYPEX_SLIDER	259
9.29.2	Typedef Documentation	259
9.29.2.1	GSLC_CB_XSLIDER_POS	259
9.29.3	Function Documentation	259
9.29.3.1	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)	259
9.29.3.2	gslc_ElemXSliderDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↵ Redraw)	260
9.29.3.3	gslc_ElemXSliderGetPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	260
9.29.3.4	gslc_ElemXSliderSetPos(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, int16_t nPos)	260
9.29.3.5	gslc_ElemXSliderSetPosFunc(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, GSLC_CB_XSLIDER_POS funcCb)	261
9.29.3.6	gslc_ElemXSliderSetStyle(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bTrim, gslc_tsColor colTrim, uint16_t nTickDiv, int16_t nTickLen, gslc_tsColor colTick)	261
9.29.3.7	gslc_ElemXSliderTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)	261
9.30	src/elem/XSpinner.c File Reference	262
9.30.1	Variable Documentation	262
9.30.1.1	ERRSTR_NULL	262
9.30.1.2	ERRSTR_PXD_NULL	262
9.31	src/elem/XSpinner.h File Reference	263
9.32	src/elem/XTemplate.c File Reference	263
9.32.1	Function Documentation	264
9.32.1.1	gslc_ElemXTemplateCreate(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsXTemplate *pXData, gslc_tsRect rElem, char *pStrBuf, uint8_t nStrBuf↵ Max, int16_t nFontId)	264

9.32.1.2	<code>gslc_ElemXTemplateDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	265
9.32.1.3	<code>gslc_ElemXTemplateTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)</code> . . . . .	265
9.32.2	Variable Documentation . . . . .	265
9.32.2.1	<code>ERRSTR_NULL</code> . . . . .	265
9.32.2.2	<code>ERRSTR_PXD_NULL</code> . . . . .	265
9.33	<code>src/elem/XTemplate.h</code> File Reference . . . . .	266
9.33.1	Macro Definition Documentation . . . . .	267
9.33.1.1	<code>GSLC_TYPEX_TEMPLATE</code> . . . . .	267
9.33.2	Function Documentation . . . . .	267
9.33.2.1	<code>gslc_ElemXTemplateCreate(gslc_tsGui *pGui, int16_t nElemId, int16_t nPage, gslc_tsXTemplate *pXData, gslc_tsRect rElem, char *pStrBuf, uint8_t nStrBuf↵Max, int16_t nFontId)</code> . . . . .	267
9.33.2.2	<code>gslc_ElemXTemplateDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)</code> . . . . .	267
9.33.2.3	<code>gslc_ElemXTemplateTouch(void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nRelX, int16_t nRelY)</code> . . . . .	267
9.34	<code>src/elem/XTextbox.c</code> File Reference . . . . .	268
9.34.1	Function Documentation . . . . .	269
9.34.1.1	<code>gslc_ElemXTextboxAdd(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, char *pTxt)</code> 269	269
9.34.1.2	<code>gslc_ElemXTextboxBufAdd(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, unsigned char chNew, bool bAdvance)</code> . . . . .	269
9.34.1.3	<code>gslc_ElemXTextboxColReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . . .	269
9.34.1.4	<code>gslc_ElemXTextboxColSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc↵_tsColor nCol)</code> . . . . .	270
9.34.1.5	<code>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)</code> . . . . .	270
9.34.1.6	<code>gslc_ElemXTextboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType e↵Redraw)</code> . . . . .	271
9.34.1.7	<code>gslc_ElemXTextboxLineWrAdv(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . .	271
9.34.1.8	<code>gslc_ElemXTextboxReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)</code> . . . .	271
9.34.1.9	<code>gslc_ElemXTextboxScrollSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)</code> . . . . .	271

9.34.1.10	gslc_ElemXTextboxWrapSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bWrapEn)	272
9.34.2	Variable Documentation	272
9.34.2.1	ERRSTR_NULL	272
9.34.2.2	ERRSTR_PXD_NULL	272
9.35	src/elem/XTextbox.h File Reference	272
9.35.1	Macro Definition Documentation	274
9.35.1.1	GSLC_TYPEX_TEXTBOX	274
9.35.1.2	GSLC_XTEXTBOX_CODE_COL_RESET	274
9.35.1.3	GSLC_XTEXTBOX_CODE_COL_SET	274
9.35.1.4	XTEXTBOX_REDRAW_ALL	274
9.35.1.5	XTEXTBOX_REDRAW_NONE	274
9.35.2	Function Documentation	274
9.35.2.1	gslc_ElemXTextboxAdd(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, char *pTxt)	274
9.35.2.2	gslc_ElemXTextboxColReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	274
9.35.2.3	gslc_ElemXTextboxColSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, gslc_tsColor nCol)	275
9.35.2.4	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)	275
9.35.2.5	gslc_ElemXTextboxDraw(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)	276
9.35.2.6	gslc_ElemXTextboxReset(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef)	276
9.35.2.7	gslc_ElemXTextboxScrollSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, uint8_t nScrollPos, uint8_t nScrollMax)	276
9.35.2.8	gslc_ElemXTextboxWrapSet(gslc_tsGui *pGui, gslc_tsElemRef *pElemRef, bool bWrapEn)	277
9.36	src/GUISlice.c File Reference	277
9.36.1	Enumeration Type Documentation	285
9.36.1.1	gslc_teDebugPrintState	285
9.36.2	Function Documentation	285
9.36.2.1	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)	285

9.36.2.2	<code>gslc_FontSetBase(gslc_tsGui *pGui, uint8_t nFontInd, int16_t nFontId, gslc_t↵ FontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code>	285
9.36.2.3	<code>gslc_OrderCoord(int16_t *pnX0, int16_t *pnY0, int16_t *pnX1, int16_t *pnY1)</code>	285
9.36.2.4	<code>gslc_SwapCoords(int16_t *pnXa, int16_t *pnYa, int16_t *pnXb, int16_t *pnYb)</code>	285
9.36.3	Variable Documentation	285
9.36.3.1	<code>ERRSTR_NULL</code>	286
9.36.3.2	<code>ERRSTR_PXD_NULL</code>	286
9.36.3.3	<code>g_pfDebugOut</code>	286
9.36.3.4	<code>m_nLUTSinF0X16</code>	286
9.37	<code>src/GUIslice.h</code> File Reference	286
9.37.1	Macro Definition Documentation	300
9.37.1.1	<code>GSLC_2PI</code>	300
9.37.1.2	<code>GSLC_ALIGN_BOT_LEFT</code>	300
9.37.1.3	<code>GSLC_ALIGN_BOT_MID</code>	300
9.37.1.4	<code>GSLC_ALIGN_BOT_RIGHT</code>	300
9.37.1.5	<code>GSLC_ALIGN_MID_LEFT</code>	300
9.37.1.6	<code>GSLC_ALIGN_MID_MID</code>	300
9.37.1.7	<code>GSLC_ALIGN_MID_RIGHT</code>	300
9.37.1.8	<code>GSLC_ALIGN_TOP_LEFT</code>	301
9.37.1.9	<code>GSLC_ALIGN_TOP_MID</code>	301
9.37.1.10	<code>GSLC_ALIGN_TOP_RIGHT</code>	301
9.37.1.11	<code>GSLC_ALIGHN_LEFT</code>	301
9.37.1.12	<code>GSLC_ALIGHN_MID</code>	301
9.37.1.13	<code>GSLC_ALIGHN_RIGHT</code>	301
9.37.1.14	<code>GSLC_ALIGNV_BOT</code>	301
9.37.1.15	<code>GSLC_ALIGNV_MID</code>	301
9.37.1.16	<code>GSLC_ALIGNV_TOP</code>	301
9.37.1.17	<code>GSLC_COL_BLACK</code>	301
9.37.1.18	<code>GSLC_COL_BLUE</code>	302
9.37.1.19	<code>GSLC_COL_BLUE_DK1</code>	302
9.37.1.20	<code>GSLC_COL_BLUE_DK2</code>	302



9.37.1.21 GSLC_COL_BLUE_DK3 . . . . .	302
9.37.1.22 GSLC_COL_BLUE_DK4 . . . . .	302
9.37.1.23 GSLC_COL_BLUE_LT1 . . . . .	302
9.37.1.24 GSLC_COL_BLUE_LT2 . . . . .	302
9.37.1.25 GSLC_COL_BLUE_LT3 . . . . .	302
9.37.1.26 GSLC_COL_BLUE_LT4 . . . . .	302
9.37.1.27 GSLC_COL_BROWN . . . . .	302
9.37.1.28 GSLC_COL_CYAN . . . . .	303
9.37.1.29 GSLC_COL_GRAY . . . . .	303
9.37.1.30 GSLC_COL_GRAY_DK1 . . . . .	303
9.37.1.31 GSLC_COL_GRAY_DK2 . . . . .	303
9.37.1.32 GSLC_COL_GRAY_DK3 . . . . .	303
9.37.1.33 GSLC_COL_GRAY_LT1 . . . . .	303
9.37.1.34 GSLC_COL_GRAY_LT2 . . . . .	303
9.37.1.35 GSLC_COL_GRAY_LT3 . . . . .	303
9.37.1.36 GSLC_COL_GREEN . . . . .	303
9.37.1.37 GSLC_COL_GREEN_DK1 . . . . .	303
9.37.1.38 GSLC_COL_GREEN_DK2 . . . . .	304
9.37.1.39 GSLC_COL_GREEN_DK3 . . . . .	304
9.37.1.40 GSLC_COL_GREEN_DK4 . . . . .	304
9.37.1.41 GSLC_COL_GREEN_LT1 . . . . .	304
9.37.1.42 GSLC_COL_GREEN_LT2 . . . . .	304
9.37.1.43 GSLC_COL_GREEN_LT3 . . . . .	304
9.37.1.44 GSLC_COL_GREEN_LT4 . . . . .	304
9.37.1.45 GSLC_COL_MAGENTA . . . . .	304
9.37.1.46 GSLC_COL_ORANGE . . . . .	304
9.37.1.47 GSLC_COL_PURPLE . . . . .	304
9.37.1.48 GSLC_COL_RED . . . . .	305
9.37.1.49 GSLC_COL_RED_DK1 . . . . .	305
9.37.1.50 GSLC_COL_RED_DK2 . . . . .	305

9.37.1.51	GSLC_COL_RED_DK3	305
9.37.1.52	GSLC_COL_RED_DK4	305
9.37.1.53	GSLC_COL_RED_LT1	305
9.37.1.54	GSLC_COL_RED_LT2	305
9.37.1.55	GSLC_COL_RED_LT3	305
9.37.1.56	GSLC_COL_RED_LT4	305
9.37.1.57	GSLC_COL_TEAL	305
9.37.1.58	GSLC_COL_WHITE	306
9.37.1.59	GSLC_COL_YELLOW	306
9.37.1.60	GSLC_COL_YELLOW_DK	306
9.37.1.61	GSLC_COLMONO_BLACK	306
9.37.1.62	GSLC_COLMONO_WHITE	306
9.37.1.63	GSLC_ELEM_FEA_CLICK_EN	306
9.37.1.64	GSLC_ELEM_FEA_FILL_EN	306
9.37.1.65	GSLC_ELEM_FEA_FRAME_EN	306
9.37.1.66	GSLC_ELEM_FEA_GLOW_EN	306
9.37.1.67	GSLC_ELEM_FEA_NONE	306
9.37.1.68	GSLC_ELEM_FEA_ROUND_EN	307
9.37.1.69	GSLC_ELEM_FEA_VALID	307
9.37.1.70	GSLC_ELEMREF_DEFAULT	307
9.37.1.71	GSLC_PMEM	307
9.37.2	Typedef Documentation	307
9.37.2.1	GSLC_CB_DEBUG_OUT	307
9.37.2.2	GSLC_CB_DRAW	307
9.37.2.3	GSLC_CB_EVENT	307
9.37.2.4	GSLC_CB_INPUT	307
9.37.2.5	GSLC_CB_PIN_POLL	307
9.37.2.6	GSLC_CB_TICK	307
9.37.2.7	GSLC_CB_TOUCH	308
9.37.2.8	gslc_tsColor	308

9.37.2.9	<a href="#">gslc_tsElem</a>	308
9.37.2.10	<a href="#">gslc_tsEvent</a>	308
9.37.2.11	<a href="#">gslc_tsEventTouch</a>	308
9.37.2.12	<a href="#">gslc_tsPt</a>	308
9.37.2.13	<a href="#">gslc_tsRect</a>	308
9.37.3	<a href="#">Enumeration Type Documentation</a>	309
9.37.3.1	<a href="#">gslc_teAction</a>	309
9.37.3.2	<a href="#">gslc_teElemId</a>	309
9.37.3.3	<a href="#">gslc_teElemInd</a>	309
9.37.3.4	<a href="#">gslc_teElemRefFlags</a>	310
9.37.3.5	<a href="#">gslc_teEventSubType</a>	310
9.37.3.6	<a href="#">gslc_teEventType</a>	310
9.37.3.7	<a href="#">gslc_teFontId</a>	311
9.37.3.8	<a href="#">gslc_teFontRefMode</a>	311
9.37.3.9	<a href="#">gslc_teFontRefType</a>	311
9.37.3.10	<a href="#">gslc_teGroupId</a>	312
9.37.3.11	<a href="#">gslc_telmgRefFlags</a>	312
9.37.3.12	<a href="#">gslc_telnitStat</a>	312
9.37.3.13	<a href="#">gslc_telInputRawEvent</a>	312
9.37.3.14	<a href="#">gslc_tePageId</a>	313
9.37.3.15	<a href="#">gslc_tePin</a>	313
9.37.3.16	<a href="#">gslc_teRedrawType</a>	313
9.37.3.17	<a href="#">gslc_teStackPage</a>	314
9.37.3.18	<a href="#">gslc_teTouch</a>	314
9.37.3.19	<a href="#">gslc_teTxtFlags</a>	315
9.37.3.20	<a href="#">gslc_teTypeCore</a>	315
9.37.4	<a href="#">Variable Documentation</a>	315
9.37.4.1	<a href="#">g_pfDebugOut</a>	315
9.38	<a href="#">src/GUISlice_config.h File Reference</a>	316
9.39	<a href="#">src/GUISlice_config_ard.h File Reference</a>	316

9.39.1	Macro Definition Documentation	317
9.39.1.1	ADAGFX_PIN_CLK	317
9.39.1.2	ADAGFX_PIN_CS	317
9.39.1.3	ADAGFX_PIN_DC	317
9.39.1.4	ADAGFX_PIN_MISO	317
9.39.1.5	ADAGFX_PIN_MOSI	317
9.39.1.6	ADAGFX_PIN_RD	317
9.39.1.7	ADAGFX_PIN_RST	317
9.39.1.8	ADAGFX_PIN_SDCS	317
9.39.1.9	ADAGFX_PIN_WR	317
9.39.1.10	ADAGFX_SPI_HW	317
9.39.1.11	ADATOUCH_FLIP_X	317
9.39.1.12	ADATOUCH_FLIP_Y	317
9.39.1.13	ADATOUCH_SWAP_XY	317
9.39.1.14	DEBUG_ERR	317
9.39.1.15	DRV_DISP_ADAGFX	317
9.39.1.16	DRV_DISP_ADAGFX_ILI9341	317
9.39.1.17	DRV_TOUCH_NONE	317
9.39.1.18	GSLC_BMP_TRANS_EN	317
9.39.1.19	GSLC_BMP_TRANS_RGB	317
9.39.1.20	GSLC_CLIP_EN	317
9.39.1.21	GSLC_DEV_TOUCH	317
9.39.1.22	GSLC_FEATURE_COMPOUND	317
9.39.1.23	GSLC_FEATURE_INPUT	318
9.39.1.24	GSLC_FEATURE_XGAUGE_RADIAL	318
9.39.1.25	GSLC_FEATURE_XGAUGE_RAMP	318
9.39.1.26	GSLC_FEATURE_XTEXTBOX_EMBED	318
9.39.1.27	GSLC_LOCAL_STR	318
9.39.1.28	GSLC_LOCAL_STR_LEN	318
9.39.1.29	GSLC_ROTATE	318

9.39.1.30	GSLC_SD_BUFFPIXEL	318
9.39.1.31	GSLC_SD_EN	318
9.39.1.32	GSLC_TOUCH_MAX_EVT	318
9.39.1.33	GSLC_USE_FLOAT	318
9.39.1.34	GSLC_USE_PROGMEM	318
9.39.1.35	TOUCH_ROTATION_DATA	318
9.39.1.36	TOUCH_ROTATION_FLIPX	318
9.39.1.37	TOUCH_ROTATION_FLIPY	318
9.39.1.38	TOUCH_ROTATION_SWAPXY	318
9.40	src/GUIslice_config_linux.h File Reference	318
9.40.1	Macro Definition Documentation	319
9.40.1.1	ADATOUCH_FLIP_X	319
9.40.1.2	ADATOUCH_FLIP_Y	319
9.40.1.3	ADATOUCH_SWAP_XY	319
9.40.1.4	DEBUG_ERR	319
9.40.1.5	DRV_DISP_SDL1	319
9.40.1.6	DRV_SDL_FIX_START	319
9.40.1.7	DRV_SDL_MOUSE_SHOW	319
9.40.1.8	DRV_TOUCH_TSLIB	319
9.40.1.9	GSLC_BMP_TRANS_EN	319
9.40.1.10	GSLC_BMP_TRANS_RGB	319
9.40.1.11	GSLC_DEV_FB	319
9.40.1.12	GSLC_DEV_TOUCH	319
9.40.1.13	GSLC_DEV_VID_DRV	319
9.40.1.14	GSLC_FEATURE_COMPOUND	319
9.40.1.15	GSLC_FEATURE_INPUT	319
9.40.1.16	GSLC_FEATURE_XGAUGE_RADIAL	319
9.40.1.17	GSLC_FEATURE_XGAUGE_RAMP	319
9.40.1.18	GSLC_FEATURE_XTEXTBOX_EMBED	319
9.40.1.19	GSLC_LOCAL_STR	319

9.40.1.20	GSLC_LOCAL_STR_LEN . . . . .	319
9.40.1.21	GSLC_TOUCH_MAX_EVT . . . . .	319
9.40.1.22	GSLC_USE_FLOAT . . . . .	319
9.40.1.23	GSLC_USE_PROGMEM . . . . .	319
9.41	src/GUISlice_drv.h File Reference . . . . .	319
9.42	src/GUISlice_drv_adagfx.cpp File Reference . . . . .	320
9.43	src/GUISlice_drv_adagfx.h File Reference . . . . .	320
9.43.1	Detailed Description . . . . .	323
9.43.2	Macro Definition Documentation . . . . .	323
9.43.2.1	DRV_HAS_DRAW_CIRCLE_FILL . . . . .	323
9.43.2.2	DRV_HAS_DRAW_CIRCLE_FRAME . . . . .	323
9.43.2.3	DRV_HAS_DRAW_LINE . . . . .	323
9.43.2.4	DRV_HAS_DRAW_POINT . . . . .	323
9.43.2.5	DRV_HAS_DRAW_POINTS . . . . .	323
9.43.2.6	DRV_HAS_DRAW_RECT_FILL . . . . .	323
9.43.2.7	DRV_HAS_DRAW_RECT_FRAME . . . . .	323
9.43.2.8	DRV_HAS_DRAW_RECT_ROUND_FILL . . . . .	324
9.43.2.9	DRV_HAS_DRAW_RECT_ROUND_FRAME . . . . .	324
9.43.2.10	DRV_HAS_DRAW_TEXT . . . . .	324
9.43.2.11	DRV_HAS_DRAW_TRI_FILL . . . . .	324
9.43.2.12	DRV_HAS_DRAW_TRI_FRAME . . . . .	324
9.43.2.13	DRV_OVERRIDE_TXT_ALIGN . . . . .	324
9.43.3	Function Documentation . . . . .	324
9.43.3.1	gslc_DrvAdaptColorToRaw(gslc_tsColor nCol) . . . . .	324
9.43.3.2	gslc_DrvDestruct(gslc_tsGui *pGui) . . . . .	324
9.43.3.3	gslc_DrvDrawBgnd(gslc_tsGui *pGui) . . . . .	325
9.43.3.4	gslc_DrvDrawBmp24FromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem) . . . . .	325
9.43.3.5	gslc_DrvDrawFillCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol) . . . . .	325
9.43.3.6	gslc_DrvDrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol) . . . . .	326

9.43.3.7	<code>gslc_DrvDrawFillRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</code>	326
9.43.3.8	<code>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)</code>	326
9.43.3.9	<code>gslc_DrvDrawFrameCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</code>	327
9.43.3.10	<code>gslc_DrvDrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code>	327
9.43.3.11	<code>gslc_DrvDrawFrameRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</code>	327
9.43.3.12	<code>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)</code>	328
9.43.3.13	<code>gslc_DrvDrawImage(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, gslc_tsImgRef sImgRef)</code>	328
9.43.3.14	<code>gslc_DrvDrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</code>	328
9.43.3.15	<code>gslc_DrvDrawMonoFromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem)</code>	329
9.43.3.16	<code>gslc_DrvDrawPoint(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code>	329
9.43.3.17	<code>gslc_DrvDrawPoints(gslc_tsGui *pGui, gslc_tsPt *asPt, uint16_t nNumPt, gslc_tsColor nCol)</code>	330
9.43.3.18	<code>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)</code>	330
9.43.3.19	<code>gslc_DrvFontAdd(gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code>	330
9.43.3.20	<code>gslc_DrvFontsDestruct(gslc_tsGui *pGui)</code>	331
9.43.3.21	<code>gslc_DrvGetDriverDisp(gslc_tsGui *pGui)</code>	331
9.43.3.22	<code>gslc_DrvGetDriverTouch(gslc_tsGui *pGui)</code>	331
9.43.3.23	<code>gslc_DrvGetNameDisp(gslc_tsGui *pGui)</code>	332
9.43.3.24	<code>gslc_DrvGetNameTouch(gslc_tsGui *pGui)</code>	332
9.43.3.25	<code>gslc_DrvGetTouch(gslc_tsGui *pGui, int16_t *pnX, int16_t *pnY, uint16_t *pnPress, gslc_teInputRawEvent *peInputEvent, int16_t *pnInputVal)</code>	332
9.43.3.26	<code>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)</code>	333
9.43.3.27	<code>gslc_DrvImageDestruct(void *pvImg)</code>	333
9.43.3.28	<code>gslc_DrvInit(gslc_tsGui *pGui)</code>	333

9.43.3.29	<code>gslc_DrvInitTouch(gslc_tsGui *pGui, const char *acDev)</code>	334
9.43.3.30	<code>gslc_DrvInitTs(gslc_tsGui *pGui, const char *acDev)</code>	334
9.43.3.31	<code>gslc_DrvLoadImage(gslc_tsGui *pGui, gslc_tsImgRef slmgRef)</code>	334
9.43.3.32	<code>gslc_DrvPageFlipNow(gslc_tsGui *pGui)</code>	335
9.43.3.33	<code>gslc_DrvRotate(gslc_tsGui *pGui, uint8_t nRotation)</code>	335
9.43.3.34	<code>gslc_DrvSetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	335
9.43.3.35	<code>gslc_DrvSetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef slmgRef)</code>	336
9.43.3.36	<code>gslc_DrvSetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	336
9.43.3.37	<code>gslc_DrvSetElemImageGlow(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_ts← ImgRef slmgRef)</code>	336
9.43.3.38	<code>gslc_DrvSetElemImageNorm(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_ts← ImgRef slmgRef)</code>	337
9.44	<code>src/GUIslice_drv_m5stack.cpp</code> File Reference	337
9.45	<code>src/GUIslice_drv_m5stack.h</code> File Reference	337
9.45.1	Detailed Description	340
9.45.2	Macro Definition Documentation	340
9.45.2.1	<code>DRV_HAS_DRAW_CIRCLE_FILL</code>	340
9.45.2.2	<code>DRV_HAS_DRAW_CIRCLE_FRAME</code>	341
9.45.2.3	<code>DRV_HAS_DRAW_LINE</code>	341
9.45.2.4	<code>DRV_HAS_DRAW_POINT</code>	341
9.45.2.5	<code>DRV_HAS_DRAW_POINTS</code>	341
9.45.2.6	<code>DRV_HAS_DRAW_RECT_FILL</code>	341
9.45.2.7	<code>DRV_HAS_DRAW_RECT_FRAME</code>	341
9.45.2.8	<code>DRV_HAS_DRAW_RECT_ROUND_FILL</code>	341
9.45.2.9	<code>DRV_HAS_DRAW_RECT_ROUND_FRAME</code>	341
9.45.2.10	<code>DRV_HAS_DRAW_TEXT</code>	341
9.45.2.11	<code>DRV_HAS_DRAW_TRI_FILL</code>	341
9.45.2.12	<code>DRV_HAS_DRAW_TRI_FRAME</code>	342
9.45.2.13	<code>DRV_OVERRIDE_TXT_ALIGN</code>	342
9.45.3	Function Documentation	342
9.45.3.1	<code>gslc_DrvAdaptColorToRaw(gslc_tsColor nCol)</code>	342



9.45.3.2	<code>gslc_DrvDestruct(gslc_tsGui *pGui)</code> . . . . .	342
9.45.3.3	<code>gslc_DrvDrawBkgnd(gslc_tsGui *pGui)</code> . . . . .	342
9.45.3.4	<code>gslc_DrvDrawBmp24FromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem)</code> . . . . .	342
9.45.3.5	<code>gslc_DrvDrawFillCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</code> . . . . .	343
9.45.3.6	<code>gslc_DrvDrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code> .	343
9.45.3.7	<code>gslc_DrvDrawFillRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</code> . . . . .	344
9.45.3.8	<code>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)</code> . . . . .	344
9.45.3.9	<code>gslc_DrvDrawFrameCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</code> . . . . .	344
9.45.3.10	<code>gslc_DrvDrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code>	345
9.45.3.11	<code>gslc_DrvDrawFrameRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t nRadius, gslc_tsColor nCol)</code> . . . . .	345
9.45.3.12	<code>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)</code> . . . . .	345
9.45.3.13	<code>gslc_DrvDrawImage(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, gslc_tsImgRef sImgRef)</code> . . . . .	346
9.45.3.14	<code>gslc_DrvDrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</code> . . . . .	346
9.45.3.15	<code>gslc_DrvDrawMonoFromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem)</code> . . . . .	347
9.45.3.16	<code>gslc_DrvDrawPoint(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code> .	347
9.45.3.17	<code>gslc_DrvDrawPoints(gslc_tsGui *pGui, gslc_tsPt *asPt, uint16_t nNumPt, gslc_tsColor nCol)</code> . . . . .	347
9.45.3.18	<code>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)</code> . . . . .	348
9.45.3.19	<code>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)</code> . . . . .	348
9.45.3.20	<code>gslc_DrvFontAdd(gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code> . . . . .	349
9.45.3.21	<code>gslc_DrvFontsDestruct(gslc_tsGui *pGui)</code> . . . . .	349
9.45.3.22	<code>gslc_DrvGetDriverDisp(gslc_tsGui *pGui)</code> . . . . .	349
9.45.3.23	<code>gslc_DrvGetDriverTouch(gslc_tsGui *pGui)</code> . . . . .	350

9.45.3.24	<code>gslc_DrvGetNameDisp(gslc_tsGui *pGui)</code>	350
9.45.3.25	<code>gslc_DrvGetNameTouch(gslc_tsGui *pGui)</code>	350
9.45.3.26	<code>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)</code>	350
9.45.3.27	<code>gslc_DrvImageDestruct(void *pvImg)</code>	351
9.45.3.28	<code>gslc_DrvInit(gslc_tsGui *pGui)</code>	351
9.45.3.29	<code>gslc_DrvInitTs(gslc_tsGui *pGui, const char *acDev)</code>	352
9.45.3.30	<code>gslc_DrvLoadImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	352
9.45.3.31	<code>gslc_DrvPageFlipNow(gslc_tsGui *pGui)</code>	352
9.45.3.32	<code>gslc_DrvRotate(gslc_tsGui *pGui, uint8_t nRotation)</code>	353
9.45.3.33	<code>gslc_DrvSetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	353
9.45.3.34	<code>gslc_DrvSetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	353
9.45.3.35	<code>gslc_DrvSetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	354
9.45.3.36	<code>gslc_DrvSetElemImageGlow(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	354
9.45.3.37	<code>gslc_DrvSetElemImageNorm(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	354
9.45.4	Variable Documentation	354
9.45.4.1	<code>ERRSTR_NULL</code>	354
9.45.4.2	<code>ERRSTR_PXD_NULL</code>	355
9.46	<code>src/GUISlice_drv_sdl.c</code> File Reference	355
9.47	<code>src/GUISlice_drv_sdl.h</code> File Reference	355
9.47.1	Detailed Description	357
9.47.2	Macro Definition Documentation	357
9.47.2.1	<code>DRV_HAS_DRAW_POINT</code>	357
9.47.2.2	<code>DRV_OVERRIDE_TXT_ALIGN</code>	357
9.47.3	Function Documentation	357
9.47.3.1	<code>gslc_DrvAdaptColor(gslc_tsColor sCol)</code>	357
9.47.3.2	<code>gslc_DrvAdaptRect(gslc_tsRect rRect)</code>	358
9.47.3.3	<code>gslc_DrvCleanStart(const char *sTTY)</code>	358
9.47.3.4	<code>gslc_DrvDestruct(gslc_tsGui *pGui)</code>	358

9.47.3.5	<code>gslc_DrvDrawBkgnd(gslc_tsGui *pGui)</code> . . . . .	359
9.47.3.6	<code>gslc_DrvDrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code> . . . . .	359
9.47.3.7	<code>gslc_DrvDrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code> . . . . .	359
9.47.3.8	<code>gslc_DrvDrawImage(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, gslc_ts← ImgRef sImgRef)</code> . . . . .	359
9.47.3.9	<code>gslc_DrvDrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</code> . . . . .	360
9.47.3.10	<code>gslc_DrvDrawPoint(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code> . . . . .	360
9.47.3.11	<code>gslc_DrvDrawPoints(gslc_tsGui *pGui, gslc_tsPt *asPt, uint16_t nNumPt, gslc← _tsColor nCol)</code> . . . . .	361
9.47.3.12	<code>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)</code> . . . . .	362
9.47.3.13	<code>gslc_DrvFontAdd(gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code> . . . . .	362
9.47.3.14	<code>gslc_DrvFontsDestruct(gslc_tsGui *pGui)</code> . . . . .	363
9.47.3.15	<code>gslc_DrvGetDriverDisp(gslc_tsGui *pGui)</code> . . . . .	363
9.47.3.16	<code>gslc_DrvGetDriverTouch(gslc_tsGui *pGui)</code> . . . . .	363
9.47.3.17	<code>gslc_DrvGetNameDisp(gslc_tsGui *pGui)</code> . . . . .	364
9.47.3.18	<code>gslc_DrvGetNameTouch(gslc_tsGui *pGui)</code> . . . . .	365
9.47.3.19	<code>gslc_DrvGetTouch(gslc_tsGui *pGui, int16_t *pnX, int16_t *pnY, uint16_t *pn← Press, gslc_teInputRawEvent *peInputEvent, int16_t *pnInputVal)</code> . . . . .	365
9.47.3.20	<code>gslc_DrvGetTxtSize(gslc_tsGui *pGui, gslc_tsFont *pFont, const char *pStr, gslc_teTxtFlags eTxtFlags, int16_t *pnTxtX, int16_t *pnTxtY, uint16_t *pnTxt← SzW, uint16_t *pnTxtSzH)</code> . . . . .	365
9.47.3.21	<code>gslc_DrvImageDestruct(void *pvImg)</code> . . . . .	366
9.47.3.22	<code>gslc_DrvInit(gslc_tsGui *pGui)</code> . . . . .	366
9.47.3.23	<code>gslc_DrvInitTouch(gslc_tsGui *pGui, const char *acDev)</code> . . . . .	367
9.47.3.24	<code>gslc_DrvLoadImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code> . . . . .	367
9.47.3.25	<code>gslc_DrvPageFlipNow(gslc_tsGui *pGui)</code> . . . . .	367
9.47.3.26	<code>gslc_DrvReportInfoPost()</code> . . . . .	368
9.47.3.27	<code>gslc_DrvReportInfoPre()</code> . . . . .	368
9.47.3.28	<code>gslc_DrvRotate(gslc_tsGui *pGui, uint8_t nRotation)</code> . . . . .	368
9.47.3.29	<code>gslc_DrvSetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code> . . . . .	368

9.47.3.30	<code>gslc_DrvSetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	369
9.47.3.31	<code>gslc_DrvSetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	369
9.47.3.32	<code>gslc_DrvSetElemImageGlow(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	369
9.47.3.33	<code>gslc_DrvSetElemImageNorm(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	369
9.48	<code>src/GUISlice_drv_tft_espi.cpp</code> File Reference	370
9.49	<code>src/GUISlice_drv_tft_espi.h</code> File Reference	370
9.49.1	Detailed Description	373
9.49.2	Macro Definition Documentation	373
9.49.2.1	<code>DRV_HAS_DRAW_CIRCLE_FILL</code>	373
9.49.2.2	<code>DRV_HAS_DRAW_CIRCLE_FRAME</code>	373
9.49.2.3	<code>DRV_HAS_DRAW_LINE</code>	373
9.49.2.4	<code>DRV_HAS_DRAW_POINT</code>	373
9.49.2.5	<code>DRV_HAS_DRAW_POINTS</code>	373
9.49.2.6	<code>DRV_HAS_DRAW_RECT_FILL</code>	374
9.49.2.7	<code>DRV_HAS_DRAW_RECT_FRAME</code>	374
9.49.2.8	<code>DRV_HAS_DRAW_RECT_ROUND_FILL</code>	374
9.49.2.9	<code>DRV_HAS_DRAW_RECT_ROUND_FRAME</code>	374
9.49.2.10	<code>DRV_HAS_DRAW_TEXT</code>	374
9.49.2.11	<code>DRV_HAS_DRAW_TRI_FILL</code>	374
9.49.2.12	<code>DRV_HAS_DRAW_TRI_FRAME</code>	374
9.49.2.13	<code>DRV_OVERRIDE_TXT_ALIGN</code>	374
9.49.3	Function Documentation	374
9.49.3.1	<code>gslc_DrvAdaptColorToRaw(gslc_tsColor nCol)</code>	374
9.49.3.2	<code>gslc_DrvDestruct(gslc_tsGui *pGui)</code>	374
9.49.3.3	<code>gslc_DrvDrawBkgnd(gslc_tsGui *pGui)</code>	375
9.49.3.4	<code>gslc_DrvDrawBmp24FromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem)</code>	375
9.49.3.5	<code>gslc_DrvDrawFillCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</code>	375
9.49.3.6	<code>gslc_DrvDrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code>	376

9.49.3.7	<code>gslc_DrvDrawFillRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t n← Radius, gslc_tsColor nCol)</code> . . . . .	376
9.49.3.8	<code>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)</code> . . . . .	376
9.49.3.9	<code>gslc_DrvDrawFrameCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMid← Y, uint16_t nRadius, gslc_tsColor nCol)</code> . . . . .	377
9.49.3.10	<code>gslc_DrvDrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code>	377
9.49.3.11	<code>gslc_DrvDrawFrameRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t n← Radius, gslc_tsColor nCol)</code> . . . . .	378
9.49.3.12	<code>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)</code> . . . . .	378
9.49.3.13	<code>gslc_DrvDrawImage(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, gslc_ts← ImgRef slmgRef)</code> . . . . .	378
9.49.3.14	<code>gslc_DrvDrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</code> . . . . .	379
9.49.3.15	<code>gslc_DrvDrawMonoFromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDst← Y, const unsigned char *pBitmap, bool bProgMem)</code> . . . . .	379
9.49.3.16	<code>gslc_DrvDrawPoint(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code> .	379
9.49.3.17	<code>gslc_DrvDrawPoints(gslc_tsGui *pGui, gslc_tsPt *asPt, uint16_t nNumPt, gslc← _tsColor nCol)</code> . . . . .	380
9.49.3.18	<code>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)</code> . . . . .	380
9.49.3.19	<code>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_teTxt← Flags eTxtFlags, gslc_tsColor colTxt, gslc_tsColor colBg)</code> . . . . .	381
9.49.3.20	<code>gslc_DrvFontAdd(gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code> . . . . .	381
9.49.3.21	<code>gslc_DrvFontsDestruct(gslc_tsGui *pGui)</code> . . . . .	381
9.49.3.22	<code>gslc_DrvGetDriverDisp(gslc_tsGui *pGui)</code> . . . . .	382
9.49.3.23	<code>gslc_DrvGetDriverTouch(gslc_tsGui *pGui)</code> . . . . .	382
9.49.3.24	<code>gslc_DrvGetNameDisp(gslc_tsGui *pGui)</code> . . . . .	382
9.49.3.25	<code>gslc_DrvGetNameTouch(gslc_tsGui *pGui)</code> . . . . .	383
9.49.3.26	<code>gslc_DrvGetTxtSize(gslc_tsGui *pGui, gslc_tsFont *pFont, const char *pStr, gslc_teTxtFlags eTxtFlags, int16_t *pnTxtX, int16_t *pnTxtY, uint16_t *pnTxt← SzW, uint16_t *pnTxtSzH)</code> . . . . .	383
9.49.3.27	<code>gslc_DrvImageDestruct(void *pvImg)</code> . . . . .	383
9.49.3.28	<code>gslc_DrvInit(gslc_tsGui *pGui)</code> . . . . .	384

9.49.3.29	<code>gslc_DrvInitTs(gslc_tsGui *pGui, const char *acDev)</code>	384
9.49.3.30	<code>gslc_DrvLoadImage(gslc_tsGui *pGui, gslc_tsImgRef slmgRef)</code>	385
9.49.3.31	<code>gslc_DrvPageFlipNow(gslc_tsGui *pGui)</code>	385
9.49.3.32	<code>gslc_DrvRotate(gslc_tsGui *pGui, uint8_t nRotation)</code>	385
9.49.3.33	<code>gslc_DrvSetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	385
9.49.3.34	<code>gslc_DrvSetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef slmgRef)</code>	386
9.49.3.35	<code>gslc_DrvSetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	386
9.49.3.36	<code>gslc_DrvSetElemImageGlow(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_ts↵ ImgRef slmgRef)</code>	386
9.49.3.37	<code>gslc_DrvSetElemImageNorm(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_ts↵ ImgRef slmgRef)</code>	387
9.50	<code>src/GUISlice_drv_utf.cpp</code> File Reference	387
9.51	<code>src/GUISlice_drv_utf.h</code> File Reference	387
9.51.1	Detailed Description	390
9.51.2	Macro Definition Documentation	390
9.51.2.1	<code>DRV_HAS_DRAW_CIRCLE_FILL</code>	390
9.51.2.2	<code>DRV_HAS_DRAW_CIRCLE_FRAME</code>	391
9.51.2.3	<code>DRV_HAS_DRAW_LINE</code>	391
9.51.2.4	<code>DRV_HAS_DRAW_POINT</code>	391
9.51.2.5	<code>DRV_HAS_DRAW_POINTS</code>	391
9.51.2.6	<code>DRV_HAS_DRAW_RECT_FILL</code>	391
9.51.2.7	<code>DRV_HAS_DRAW_RECT_FRAME</code>	391
9.51.2.8	<code>DRV_HAS_DRAW_RECT_ROUND_FILL</code>	391
9.51.2.9	<code>DRV_HAS_DRAW_RECT_ROUND_FRAME</code>	391
9.51.2.10	<code>DRV_HAS_DRAW_TEXT</code>	391
9.51.2.11	<code>DRV_HAS_DRAW_TRI_FILL</code>	391
9.51.2.12	<code>DRV_HAS_DRAW_TRI_FRAME</code>	392
9.51.2.13	<code>DRV_OVERRIDE_TXT_ALIGN</code>	392
9.51.3	Function Documentation	392
9.51.3.1	<code>gslc_DrvAdaptColorToRaw(gslc_tsColor nCol)</code>	392
9.51.3.2	<code>gslc_DrvDestruct(gslc_tsGui *pGui)</code>	392

9.51.3.3	<code>gslc_DrvDrawBkgnd(gslc_tsGui *pGui)</code> . . . . .	392
9.51.3.4	<code>gslc_DrvDrawBmp24FromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, const unsigned char *pBitmap, bool bProgMem)</code> . . . . .	392
9.51.3.5	<code>gslc_DrvDrawFillCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMidY, uint16_t nRadius, gslc_tsColor nCol)</code> . . . . .	393
9.51.3.6	<code>gslc_DrvDrawFillRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code> .	393
9.51.3.7	<code>gslc_DrvDrawFillRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t n← Radius, gslc_tsColor nCol)</code> . . . . .	394
9.51.3.8	<code>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)</code> . . . . .	394
9.51.3.9	<code>gslc_DrvDrawFrameCircle(gslc_tsGui *pGui, int16_t nMidX, int16_t nMid← Y, uint16_t nRadius, gslc_tsColor nCol)</code> . . . . .	394
9.51.3.10	<code>gslc_DrvDrawFrameRect(gslc_tsGui *pGui, gslc_tsRect rRect, gslc_tsColor nCol)</code>	395
9.51.3.11	<code>gslc_DrvDrawFrameRoundRect(gslc_tsGui *pGui, gslc_tsRect rRect, int16_t n← Radius, gslc_tsColor nCol)</code> . . . . .	395
9.51.3.12	<code>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)</code> . . . . .	395
9.51.3.13	<code>gslc_DrvDrawImage(gslc_tsGui *pGui, int16_t nDstX, int16_t nDstY, gslc_ts← lmgRef slmgRef)</code> . . . . .	396
9.51.3.14	<code>gslc_DrvDrawLine(gslc_tsGui *pGui, int16_t nX0, int16_t nY0, int16_t nX1, int16_t nY1, gslc_tsColor nCol)</code> . . . . .	396
9.51.3.15	<code>gslc_DrvDrawMonoFromMem(gslc_tsGui *pGui, int16_t nDstX, int16_t nDst← Y, const unsigned char *pBitmap, bool bProgMem)</code> . . . . .	397
9.51.3.16	<code>gslc_DrvDrawPoint(gslc_tsGui *pGui, int16_t nX, int16_t nY, gslc_tsColor nCol)</code> .	397
9.51.3.17	<code>gslc_DrvDrawPoints(gslc_tsGui *pGui, gslc_tsPt *asPt, uint16_t nNumPt, gslc← _tsColor nCol)</code> . . . . .	397
9.51.3.18	<code>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)</code> . . . . .	398
9.51.3.19	<code>gslc_DrvFontAdd(gslc_teFontRefType eFontRefType, const void *pvFontRef, uint16_t nFontSz)</code> . . . . .	398
9.51.3.20	<code>gslc_DrvFontsDestruct(gslc_tsGui *pGui)</code> . . . . .	399
9.51.3.21	<code>gslc_DrvGetDriverDisp(gslc_tsGui *pGui)</code> . . . . .	399
9.51.3.22	<code>gslc_DrvGetDriverTouch(gslc_tsGui *pGui)</code> . . . . .	399
9.51.3.23	<code>gslc_DrvGetNameDisp(gslc_tsGui *pGui)</code> . . . . .	400
9.51.3.24	<code>gslc_DrvGetNameTouch(gslc_tsGui *pGui)</code> . . . . .	401

9.51.3.25	<code>gslc_DrvGetTouch(gslc_tsGui *pGui, int16_t *pnX, int16_t *pnY, uint16_t *pnPress, gslc_telInputRawEvent *peInputEvent, int16_t *pnInputVal)</code>	401
9.51.3.26	<code>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)</code>	401
9.51.3.27	<code>gslc_DrvImageDestruct(void *pvImg)</code>	402
9.51.3.28	<code>gslc_DrvInit(gslc_tsGui *pGui)</code>	402
9.51.3.29	<code>gslc_DrvInitTouch(gslc_tsGui *pGui, const char *acDev)</code>	403
9.51.3.30	<code>gslc_DrvInitTs(gslc_tsGui *pGui, const char *acDev)</code>	403
9.51.3.31	<code>gslc_DrvLoadImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	403
9.51.3.32	<code>gslc_DrvPageFlipNow(gslc_tsGui *pGui)</code>	404
9.51.3.33	<code>gslc_DrvRotate(gslc_tsGui *pGui, uint8_t nRotation)</code>	404
9.51.3.34	<code>gslc_DrvSetBkgndColor(gslc_tsGui *pGui, gslc_tsColor nCol)</code>	404
9.51.3.35	<code>gslc_DrvSetBkgndImage(gslc_tsGui *pGui, gslc_tsImgRef sImgRef)</code>	404
9.51.3.36	<code>gslc_DrvSetClipRect(gslc_tsGui *pGui, gslc_tsRect *pRect)</code>	405
9.51.3.37	<code>gslc_DrvSetElemImageGlow(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	405
9.51.3.38	<code>gslc_DrvSetElemImageNorm(gslc_tsGui *pGui, gslc_tsElem *pElem, gslc_tsImgRef sImgRef)</code>	405
9.52	<code>src/GUISlice_ex.h</code> File Reference	406
9.53	<code>src/GUISlice_th.cpp</code> File Reference	406
9.53.1	Function Documentation	407
9.53.1.1	<code>gslc_getTouchHandler(void)</code>	407
9.53.1.2	<code>gslc_InitTouchHandler(TouchHandler *pTH)</code>	407
9.53.2	Variable Documentation	407
9.53.2.1	<code>pTouchHandler</code>	407
9.54	<code>src/GUISlice_th.h</code> File Reference	407
9.54.1	Function Documentation	408
9.54.1.1	<code>gslc_getTouchHandler(void)</code>	408
9.54.1.2	<code>gslc_InitTouchHandler(TouchHandler *pTHO)</code>	408
9.55	<code>src/GUISlice_th_XPT2046.h</code> File Reference	408
9.56	<code>src/GUISlice_version.h</code> File Reference	409
9.56.1	Macro Definition Documentation	409
9.56.1.1	<code>GUISLICE_VER</code>	409



# 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, keypad, listbox, radial controls, scrolling textbox / terminal, graphs, etc. plus extensions and multiple pages.
- Cross-platform **GUIslice Builder** (beta) application to generate layouts
- *Platform-independent* GUI core currently supports:
  - Adafruit-GFX, TFT\_eSPI, mcufriend, UTFT, SDL1.2, SDL2.0
- *Devices:*
  - Raspberry Pi, Arduino, ESP8266 / NodeMCU, ESP32, M5stack, Teensy 3, 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, ILI9341\_t3, ILI9341\_due
- *Touchscreen control including:*
  - STMPE610, FT6206, XPT2046, 4-wire, tslib, URTouch, Adafruit Seesaw
- Foreign characters / UTF-8 encoding (in SDL mode), anti-aliased fonts (in TFT\_eSPI mode)
- Dynamic display rotation
- GPIO / pin / keyboard / Adafruit Seesaw control 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

## Chapter 2

### Todo List

Global [gslc\\_CollectFindFocusStep](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, bool bNext, bool \*pb↵  
Wrapped, int16\_t \*pnElemInd)

Doc. This API is experimental and subject to change

Global [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)

Doc. This API is experimental and subject to change

Global [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) elInputEvent, int16\_t nInputVal, [gslc↵  
\\_teAction](#) eAction, int16\_t nActionVal)

Doc. This API is experimental and subject to change

Global [gslc\\_InputMapLookup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) elInputEvent, int16\_t nInputVal,  
[gslc\\_teAction](#) \*peAction, int16\_t \*pnActionVal)

Doc. This API is experimental and subject to change

Global [gslc\\_PageFocusStep](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage, bool bNext)

Doc. This API is experimental and subject to change

Global [gslc\\_SetPinPollFunc](#) ([gslc\\_tsGui](#) \*pGui, [GSLC\\_CB\\_PIN\\_POLL](#) pfunc)

Doc. This API is experimental and subject to change



## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

General Functions . . . . .	13
Graphics General Functions . . . . .	20
Graphics Primitive Functions . . . . .	29
Font Functions . . . . .	38
Page Functions . . . . .	41
Element Functions . . . . .	46
Element: Creation Functions . . . . .	47
Element: General Functions . . . . .	51
Element: Update Functions . . . . .	52
Touchscreen Functions . . . . .	66
Input Mapping Functions . . . . .	70
General Purpose Macros . . . . .	71
Flash-based Element Macros . . . . .	72
Internal Functions . . . . .	75
Internal: Misc Functions . . . . .	92
Internal: Element Functions . . . . .	93
Internal: Page Functions . . . . .	99
Internal: Element Collection Functions . . . . .	104
Internal: Element Collection Event Functions . . . . .	110
Internal: Tracking Functions . . . . .	112
Internal: Cleanup Functions . . . . .	114



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

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

gslc_tsCollect . . . . .	119
gslc_tsColor . . . . .	120
gslc_tsDriver . . . . .	121
gslc_tsElem . . . . .	122
gslc_tsElemRef . . . . .	124
gslc_tsEvent . . . . .	124
gslc_tsEventTouch . . . . .	125
gslc_tsFont . . . . .	125
gslc_tsGui . . . . .	126
gslc_tsImgRef . . . . .	128
gslc_tsInputMap . . . . .	129
gslc_tsPage . . . . .	130
gslc_tsPt . . . . .	131
gslc_tsRect . . . . .	131
gslc_tsXCheckbox . . . . .	132
gslc_tsXGauge . . . . .	133
gslc_tsXGlowball . . . . .	136
gslc_tsXGlowballRing . . . . .	138
gslc_tsXGraph . . . . .	139
gslc_tsXListbox . . . . .	142
gslc_tsXProgress . . . . .	145
gslc_tsXRadial . . . . .	147
gslc_tsXRamp . . . . .	150
gslc_tsXRingGauge . . . . .	151
gslc_tsXSlider . . . . .	153
gslc_tsXTemplate . . . . .	155
gslc_tsXTextbox . . . . .	155
THPoint . . . . .	158
TouchHandler . . . . .	159
TouchHandler_XPT2046 . . . . .	161





## 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 . . . . .	119
<a href="#">gslc_tsColor</a>	Color structure. Defines RGB triplet . . . . .	120
<a href="#">gslc_tsDriver</a>	. . . . .	121
<a href="#">gslc_tsElem</a>	Element Struct . . . . .	122
<a href="#">gslc_tsElemRef</a>	Element reference structure . . . . .	124
<a href="#">gslc_tsEvent</a>	Event structure . . . . .	124
<a href="#">gslc_tsEventTouch</a>	Structure used to pass touch data through event . . . . .	125
<a href="#">gslc_tsFont</a>	Font reference structure . . . . .	125
<a href="#">gslc_tsGui</a>	GUI structure . . . . .	126
<a href="#">gslc_tsImgRef</a>	Image reference structure . . . . .	128
<a href="#">gslc_tsInputMap</a>	Input mapping . . . . .	129
<a href="#">gslc_tsPage</a>	Page structure . . . . .	130
<a href="#">gslc_tsPt</a>	Define point coordinates . . . . .	131
<a href="#">gslc_tsRect</a>	Rectangular region. Defines X,Y corner coordinates plus dimensions . . . . .	131
<a href="#">gslc_tsXCheckbox</a>	Extended data for Checkbox element . . . . .	132
<a href="#">gslc_tsXGauge</a>	Extended data for Gauge element . . . . .	133
<a href="#">gslc_tsXGlowball</a>	Extended data for Slider element . . . . .	136
<a href="#">gslc_tsXGlowballRing</a>	. . . . .	138
<a href="#">gslc_tsXGraph</a>	Extended data for Graph element . . . . .	139

<a href="#">gslc_tsXListbox</a>	
Extended data for Listbox element . . . . .	142
<a href="#">gslc_tsXProgress</a>	
Extended data for Gauge element . . . . .	145
<a href="#">gslc_tsXRadial</a>	
Extended data for Gauge element . . . . .	147
<a href="#">gslc_tsXRamp</a>	
Extended data for Gauge element . . . . .	150
<a href="#">gslc_tsXRingGauge</a>	
Extended data for XRingGauge element . . . . .	151
<a href="#">gslc_tsXSlider</a>	
Extended data for Slider element . . . . .	153
<a href="#">gslc_tsXTemplate</a>	
Callback function for slider feedback . . . . .	155
<a href="#">gslc_tsXTextbox</a>	
Extended data for Textbox element . . . . .	155
<a href="#">THPoint</a> . . . . .	158
<a href="#">TouchHandler</a> . . . . .	159
<a href="#">TouchHandler_XPT2046</a> . . . . .	161

## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

src/GUISlice.c	277
src/GUISlice.h	286
src/GUISlice_config.h	316
src/GUISlice_config_ard.h	316
src/GUISlice_config_linux.h	318
src/GUISlice_drv.h	319
src/GUISlice_drv_adagfx.cpp	320
src/GUISlice_drv_adagfx.h	
GUISlice library (driver layer for Adafruit-GFX)	320
src/GUISlice_drv_m5stack.cpp	337
src/GUISlice_drv_m5stack.h	
GUISlice library (driver layer for M5stack)	337
src/GUISlice_drv_sdl.c	355
src/GUISlice_drv_sdl.h	
GUISlice library (driver layer for LINUX / SDL)	355
src/GUISlice_drv_tft_espi.cpp	370
src/GUISlice_drv_tft_espi.h	
GUISlice library (driver layer for TFT-eSPI)	370
src/GUISlice_drv_utft.cpp	387
src/GUISlice_drv_utft.h	
GUISlice library (driver layer for UTFT)	387
src/GUISlice_ex.h	406
src/GUISlice_th.cpp	406
src/GUISlice_th.h	407
src/GUISlice_th_XPT2046.h	408
src/GUISlice_version.h	409
src/elem/XCheckbox.c	163
src/elem/XCheckbox.h	167
src/elem/XGauge.c	173
src/elem/XGauge.h	178
src/elem/XGlowball.c	184
src/elem/XGlowball.h	186
src/elem/XGraph.c	189
src/elem/XGraph.h	193
src/elem/XKeyPad.c	197

src/elem/XKeyPad.h	197
src/elem/XKeyPad_Alpha.c	198
src/elem/XKeyPad_Alpha.h	198
src/elem/XKeyPad_Num.c	199
src/elem/XKeyPad_Num.h	200
src/elem/XListbox.c	201
src/elem/XListbox.h	208
src/elem/XProgress.c	216
src/elem/XProgress.h	220
src/elem/XRadial.c	224
src/elem/XRadial.h	228
src/elem/XRamp.c	233
src/elem/XRamp.h	236
src/elem/XRingGauge.c	240
src/elem/XRingGauge.h	246
src/elem/XSelNum.c	251
src/elem/XSelNum.h	252
src/elem/XSlider.c	253
src/elem/XSlider.h	257
src/elem/XSpinner.c	262
src/elem/XSpinner.h	263
src/elem/XTemplate.c	263
src/elem/XTemplate.h	266
src/elem/XTextbox.c	268
src/elem/XTextbox.h	272

## 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\\_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.*
- 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 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 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.

#### 7.1.2.3 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.

**7.1.2.4 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

**7.1.2.5 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

**7.1.2.6 char\* gslc\_GetVer ( gslc\_tsGui \* pGui )**

Get the GUIslice version number.

**Parameters**

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

**Returns**

String containing version number

### 7.1.2.7 `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

### 7.1.2.8 `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

### 7.1.2.9 `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

**7.1.2.10 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

**7.1.2.11 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

#### 7.1.2.12 `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

#### 7.1.2.13 `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

#### 7.1.2.14 `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

#### 7.1.2.15 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 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.*

### 7.2.1 Detailed Description

Helper functions that support graphics operations.

### 7.2.2 Function Documentation

**7.2.2.1** `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** `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** `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_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_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 `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 `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.
- $\text{gslc\_cosFX}(n\text{AngDeg} \cdot 64) / 32768.0 = \cos(n\text{AngDeg} \cdot 2\pi / 360)$

## 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_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_tslmgRef gslc_GetImageFromFile ( const char * pFname, gslc_telmgRefFlags 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_tslmgRef gslc_GetImageFromProg ( const unsigned char * plmgBuf, gslc_telmgRefFlags eFmt )`

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

#### Parameters

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

#### Returns

Loaded image reference

### 7.2.2.11 `gslc_tslmgRef gslc_GetImageFromRam ( unsigned char * plmgBuf, gslc_telmgRefFlags eFmt )`

Create an image reference to a bitmap in SRAM.

#### Parameters

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

#### Returns

Loaded image reference

### 7.2.2.12 `gslc_tslmgRef gslc_GetImageFromSD ( const char * pFname, gslc_telmgRefFlags 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 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 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 void gslc\_InvalidateRgnReset ( gslc\_tsGui \* *pGui* )**

Reset the invalidation region.

**Parameters**

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

**Returns**

none

**7.2.2.16 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 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 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** 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** 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** 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** 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** 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 differeng from the drawing angular range (`nAngSegStart`..`nAngSecEnd`) 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.
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 <code>nRad1</code> defines ring thickness.
in	<i>cArcStart</i>	Start color for gradient fill (with angular range defined by <code>nAngGradStart</code> , <code>nAngGradRange</code> )
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.

## Parameters

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 (cArcStart)
in	<i>nAngGradRange</i>	For gradient fill, defines the angular range associated with the start-to-end color range (cArcStart..cArcEnd)

## Returns

none

**7.3.2.3** 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** 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** 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** 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** 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



## Parameters

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** 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** 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** 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** 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** 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** 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
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** 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** 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

**Parameters**

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** 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** 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** `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_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 `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 `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 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_PageAdd (gslc_tsGui *pGui, int16_t nPageld, 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 nPageld, 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 `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 void gslc_PageAdd ( gslc_tsGui * pGui, int16_t nPageld, 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>nPageld</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_tsElemRef* gslc_PageFindElemById ( gslc_tsGui * pGui, int16_t nPageld, 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↔ Pageld</i>	Page ID to search
in	<i>n↔ ElemId</i>	Element ID to search

**Returns**

Ptr to an element or NULL if none found

**7.5.2.4** `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** `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** `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** `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↔ Pageld</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 void gslc\_SetPageBase ( gslc\_tsGui \* *pGui*, int16\_t *nPageld* )

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

**Parameters**

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

**Returns**

none

### 7.5.2.9 void gslc\_SetPageCur ( gslc\_tsGui \* *pGui*, int16\_t *nPageld* )

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

**Parameters**

in	<i>pGui</i>	Pointer to GUI
in	<i>n↔ Pageld</i>	Page ID to select

**Returns**

none

### 7.5.2.10 void gslc\_SetPageOverlay ( gslc\_tsGui \* *pGui*, int16\_t *nPageld* )

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

## Parameters

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

## Returns

none

7.5.2.11 void gslc\_SetStackPage ( gslc\_tsGui \* *pGui*, uint8\_t *nStackPos*, int16\_t *nPageld* )

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>nPageld</i>	Page ID to select as current

## Returns

none

7.5.2.12 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).

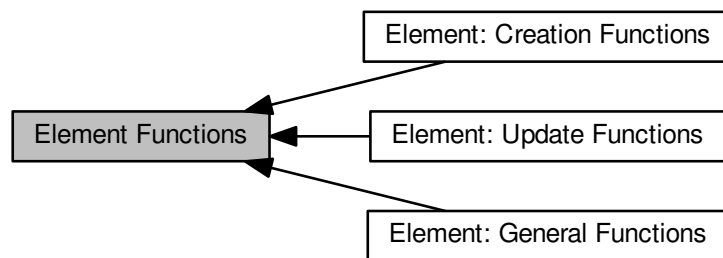
## Returns

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 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.*

### 7.7.1 Detailed Description

Functions that create GUI elements.

### 7.7.2 Function Documentation

**7.7.2.1** `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_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_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)



## Parameters

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 ( <i>pStrBuf</i> ). 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_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>n↔ElemId</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_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>n↔ElemId</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>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_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** `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

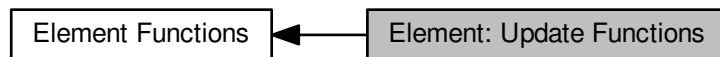
##### Returns

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\\_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\\_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.*
- 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.*
- 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 bOnlyClickEn)  
*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 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

**7.9.2.2** `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

**7.9.2.3** `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

**7.9.2.4** `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

**7.9.2.5 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

**7.9.2.6 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

**7.9.2.7 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

**7.9.2.8** `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

**7.9.2.9** `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

**7.9.2.10** `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
----	-------------	----------------



## Parameters

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

**7.9.2.11** 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

**7.9.2.12** 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

**7.9.2.13** 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

**7.9.2.14** 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

**7.9.2.15** 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

**7.9.2.16** 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

**7.9.2.17** 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

**7.9.2.18** 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

**7.9.2.19** 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

**7.9.2.20** 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

**7.9.2.21** 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

**7.9.2.22** 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

**7.9.2.23** 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

7.9.2.24 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

**7.9.2.25** 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

**7.9.2.26** 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

**7.9.2.27** 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

**7.9.2.28** 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

**7.9.2.29** 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

**7.9.2.30** 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

7.9.2.31 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

## 7.10 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 values.*
- `void gslc_SetTouchRemapYX (gslc_tsGui *pGui, bool bSwap)`  
*Configure touchscreen XY swap.*

### 7.10.1 Detailed Description

Functions that configure and respond to a touch device.

### 7.10.2 Macro Definition Documentation

#### 7.10.2.1 `#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 `#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 `#define TOUCH_ROTATION_FLIPX( rotation )`7.10.2.4 `#define TOUCH_ROTATION_FLIPX( rotation )`7.10.2.5 `#define TOUCH_ROTATION_FLIPY( rotation )`7.10.2.6 `#define TOUCH_ROTATION_FLIPY( rotation )`7.10.2.7 `#define TOUCH_ROTATION_SWAPXY( rotation )`7.10.2.8 `#define TOUCH_ROTATION_SWAPXY( rotation )`

## 7.10.3 Function Documentation

7.10.3.1 `bool gslc_GetTouch ( gslc_tsGui * pGui, int16_t * pnX, int16_t * pnY, uint16_t * pnPress, gslc_tInputRawEvent * pInputEvent, 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>pInputEvent</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 `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

**7.10.3.3** void `gslc_SetTouchRemapCal` ( `gslc_tsGui` \* *pGui*, uint16\_t *nXMin*, uint16\_t *nXMax*, uint16\_t *nYMin*, uint16\_t *nYMax* )

Configure touchscreen calibration 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

**7.10.3.4** 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

**7.10.3.5** 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)
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_teInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) eAction, int16\_t nActionVal)

### 7.11.1 Detailed Description

Functions that handle GPIO / pin and keyboard input.

### 7.11.2 Function Documentation

7.11.2.1 void [gslc\\_InitInputMap](#) ( [gslc\\_tsGui](#) \* *pGui*, [gslc\\_tsInputMap](#) \* *asInputMap*, uint8\_t *nInputMapMax* )

**Todo** Doc. This API is experimental and subject to change

7.11.2.2 void [gslc\\_InputMapAdd](#) ( [gslc\\_tsGui](#) \* *pGui*, [gslc\\_teInputRawEvent](#) *eInputEvent*, int16\_t *nInputVal*, [gslc\\_teAction](#) *eAction*, int16\_t *nActionVal* )

**Todo** Doc. This API is experimental and subject to change

7.11.2.3 void [gslc\\_SetPinPollFunc](#) ( [gslc\\_tsGui](#) \* *pGui*, [GSLC\\_CB\\_PIN\\_POLL](#) *pfunc* )

**Todo** Doc. This API is experimental and subject to change

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

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

7.12.2.3 `#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

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

7.12.2.4 `#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_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 #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 `#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
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 `#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 `#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 #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)

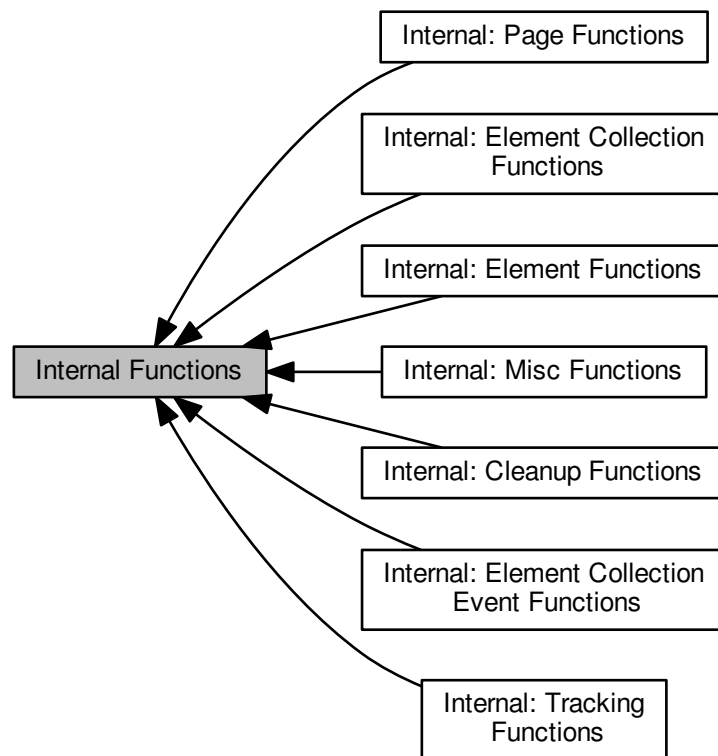
## Parameters

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

## 7.14 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_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_telmgRefFlags gslc_tsImgRef::eImgFlags`
  - Image reference flags.*
- `void * gslc_tsImgRef::pvImgRaw`
  - 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.*
- `uint8_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_tsImgRef gslc_tsElem::sImgRefNorm`  
*Image reference to draw (normal)*
- `gslc_tsImgRef gslc_tsElem::sImgRefGlow`  
*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::nElemIndFocused`
- Element index currently in focus (eg. by keyboard/pin control), 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_tsInputRawEvent gslc_tsInputMap::eEvent`
- The input event.*

  - `int16_t gslc_tsInputMap::nVal`
- The value associated with the input event.*

  - `gslc_tsAction 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 x axis.*
- `uint16_t gslc_tsGui::nTouchCalXMin`  
*Calibration X minimum reading.*
- `uint16_t gslc_tsGui::nTouchCalXMax`  
*Calibration X maximum reading.*
- `uint16_t gslc_tsGui::nTouchCalYMin`  
*Calibration Y minimum reading.*
- `uint16_t gslc_tsGui::nTouchCalYMax`  
*Calibration Y maximum reading.*
- `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::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.*
- `void * gslc_tsGui::pvDriver`  
*Driver-specific members (gslc\_tsDriver\*)*
- `bool gslc_tsGui::bRedrawPartialEn`  
*Driver supports partial page redraw.*
- `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::blInvalidateEn](#)

*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.*

### 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 bool [gslc\\_tsGui::abPageStackActive](#)[GSLC\_STACK\_\_MAX]

Whether page in stack can receive touch events.

#### 7.14.2.2 bool [gslc\\_tsGui::abPageStackDoDraw](#)[GSLC\_STACK\_\_MAX]

Whether page in stack is still actively drawn.

#### 7.14.2.3 [gslc\\_tsPage](#)\* [gslc\\_tsGui::apPageStack](#)[GSLC\_STACK\_\_MAX]

Stack of pages.



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

Array of elements.

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

Array of element references.

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

Collection of loaded fonts.

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

Array of input maps.

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

Array of all pages defined in system.

**7.14.2.9 `uint8_t` `gslc_tsColor::b`**

RGB blue value.

**7.14.2.10 `bool` `gslc_tsGui::bInvalidateEn`**

A region of the display has been invalidated.

**7.14.2.11 `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.

**7.14.2.12 `bool` `gslc_tsGui::bScreenNeedFlip`**

Screen requires a page flip.

**7.14.2.13 `bool` `gslc_tsGui::bScreenNeedRedraw`**

Screen requires a redraw.

#### 7.14.2.14 `bool gslc_tsGui::bTouchRemapEn`

Enable touch remapping?

#### 7.14.2.15 `bool gslc_tsGui::bTouchRemapYX`

Enable touch controller swapping of X & Y.

#### 7.14.2.16 `gslc_tsColor gslc_tsElem::colElemFill`

Color for background fill.

#### 7.14.2.17 `gslc_tsColor gslc_tsElem::colElemFillGlow`

Color to use for fill when glowing.

#### 7.14.2.18 `gslc_tsColor gslc_tsElem::colElemFrame`

Color for frame.

#### 7.14.2.19 `gslc_tsColor gslc_tsElem::colElemFrameGlow`

Color to use for frame when glowing.

#### 7.14.2.20 `gslc_tsColor gslc_tsElem::colElemText`

Color of overlay text.

#### 7.14.2.21 `gslc_tsColor gslc_tsElem::colElemTextGlow`

Color of overlay text when glowing.

#### 7.14.2.22 `gslc_teAction gslc_tsInputMap::eAction`

Resulting action.

#### 7.14.2.23 `gslc_teElemRefFlags gslc_tsElemRef::eElemFlags`

Element reference flags.

**7.14.2.24 gslc\_telInputRawEvent gslc\_tsInputMap::eEvent**

The input event.

**7.14.2.25 gslc\_teFontRefMode gslc\_tsFont::eFontRefMode**

Font reference mode.

**7.14.2.26 gslc\_teFontRefType gslc\_tsFont::eFontRefType**

Font reference type.

**7.14.2.27 gslc\_telmgRefFlags gslc\_tslmgRef::elmgFlags**

Image reference flags.

**7.14.2.28 gslc\_telnitStat gslc\_tsGui::elnitStatTouch**

Status of touch initialization.

**7.14.2.29 gslc\_teTouch gslc\_tsEventTouch::eTouch**

Touch state.

**7.14.2.30 int8\_t gslc\_tsElem::eTxtAlign**

Alignment of overlay text.

**7.14.2.31 gslc\_teTxtFlags gslc\_tsElem::eTxtFlags**

Flags associated with text buffer.

**7.14.2.32 gslc\_teEventType gslc\_tsEvent::eType**

Event type.

**7.14.2.33 uint8\_t gslc\_tsColor::g**

RGB green value.

#### 7.14.2.34 `uint16_t gslc_tsRect::h`

Height of region.

#### 7.14.2.35 `int16_t gslc_tsInputMap::nActionVal`

The value for the output action.

#### 7.14.2.36 `uint16_t gslc_tsGui::nDisp0H`

Height of the display (pixels) in native orientation.

#### 7.14.2.37 `uint16_t gslc_tsGui::nDisp0W`

Width of the display (pixels) in native orientation.

#### 7.14.2.38 `uint8_t gslc_tsGui::nDispDepth`

Bit depth of display (bits per pixel)

#### 7.14.2.39 `uint16_t gslc_tsGui::nDispH`

Height of the display (pixels)

#### 7.14.2.40 `uint16_t gslc_tsGui::nDispW`

Width of the display (pixels)

#### 7.14.2.41 `int16_t gslc_tsCollect::nElemAutoldNext`

Next Element ID for auto-assignment.

#### 7.14.2.42 `uint16_t gslc_tsCollect::nElemCnt`

Number of elements allocated.

#### 7.14.2.43 `int16_t gslc_tsCollect::nElemIndFocused`

Element index currently in focus (eg. by keyboard/pin control), `GSLC_IND_NONE` for none.

**7.14.2.44    uint16\_t gslc\_tsCollect::nElemMax**

Maximum number of elements to allocate (in RAM)

**7.14.2.45    uint16\_t gslc\_tsCollect::nElemRefCnt**

Number of element references allocated.

**7.14.2.46    uint16\_t gslc\_tsCollect::nElemRefMax**

Maximum number of element references to allocate.

**7.14.2.47    uint8\_t gslc\_tsElem::nFeatures**

Element feature vector (appearance/behavior))

**7.14.2.48    uint8\_t gslc\_tsGui::nFlipX**

Adafruit GFX Touch Flip x axis.

**7.14.2.49    uint8\_t gslc\_tsGui::nFlipY**

Adafruit GFX Touch Flip y axis.

**7.14.2.50    uint8\_t gslc\_tsGui::nFontCnt**

Number of fonts allocated.

**7.14.2.51    uint8\_t gslc\_tsGui::nFontMax**

Maximum number of fonts to allocate.

**7.14.2.52    uint8\_t gslc\_tsGui::nFrameRateCnt**

Diagnostic frame rate count.

**7.14.2.53    uint8\_t gslc\_tsGui::nFrameRateStart**

Diagnostic frame rate timestamp.

**7.14.2.54 int16\_t gslc\_tsElem::nGroup**

Group ID that the element belongs to.

**7.14.2.55 int16\_t gslc\_tsFont::nId**

Font ID specified by user.

**7.14.2.56 int16\_t gslc\_tsElem::nId**

Element ID specified by user.

**7.14.2.57 uint8\_t gslc\_tsGui::nInputMapCnt**

Current number of input maps.

**7.14.2.58 uint8\_t gslc\_tsGui::nInputMapMax**

Maximum number of input maps.

**7.14.2.59 uint8\_t gslc\_tsGui::nPageCnt**

Current number of pages defined.

**7.14.2.60 int16\_t gslc\_tsPage::nPageId**

Page identifier.

**7.14.2.61 uint8\_t gslc\_tsGui::nPageMax**

Maximum number of pages that can be defined.

**7.14.2.62 uint8\_t gslc\_tsGui::nRotation**

Adafruit GFX Rotation of display.

**7.14.2.63 uint8\_t gslc\_tsGui::nRoundRadius**

Radius for rounded elements.

**7.14.2.64 uint16\_t gslc\_tsFont::nSize**

Font size.

**7.14.2.65 uint8\_t gslc\_tsElem::nStrBufMax**

Size of string buffer.

**7.14.2.66 uint8\_t gslc\_tsEvent::nSubType**

Event sub-type.

**7.14.2.67 uint8\_t gslc\_tsGui::nSwapXY**

Adafruit GFX Touch Swap x and y axes.

**7.14.2.68 uint16\_t gslc\_tsGui::nTouchCalXMax**

Calibration X maximum reading.

**7.14.2.69 uint16\_t gslc\_tsGui::nTouchCalXMin**

Calibration X minimum reading.

**7.14.2.70 uint16\_t gslc\_tsGui::nTouchCalYMax**

Calibration Y maximum reading.

**7.14.2.71 uint16\_t gslc\_tsGui::nTouchCalYMin**

Calibration Y minimum reading.

**7.14.2.72 uint16\_t gslc\_tsGui::nTouchLastPress**

Last touch event pressure (0=none))

**7.14.2.73 int16\_t gslc\_tsGui::nTouchLastX**

Last touch event X coord.

**7.14.2.74 int16\_t gslc\_tsGui::nTouchLastY**

Last touch event Y coord.

**7.14.2.75 uint8\_t gslc\_tsGui::nTouchRotation**

Touchscreen rotation offset vs display.

**7.14.2.76 int8\_t gslc\_tsElem::nTxtMarginX**

Margin of overlay text within rect region (x offset)

**7.14.2.77 int8\_t gslc\_tsElem::nTxtMarginY**

Margin of overlay text within rect region (y offset)

**7.14.2.78 int16\_t gslc\_tsElem::nType**

Element type enumeration.

**7.14.2.79 int16\_t gslc\_tsInputMap::nVal**

The value associated with the input event.

**7.14.2.80 int16\_t gslc\_tsEventTouch::nX**

Touch X coordinate (or param1)

**7.14.2.81 int16\_t gslc\_tsEventTouch::nY**

Touch Y coordinate (or param2)

**7.14.2.82 gslc\_tsElem\* gslc\_tsElemRef::pElem**

Pointer to element in memory [RAM,FLASH].

**7.14.2.83 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.



**7.14.2.84 gslc\_tsElemRef\* gslc\_tsCollect::pElemRefTracked**

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

**7.14.2.85 const char\* gslc\_tsImgRef::pFname**

Pathname to input image file [FILE,SD].

**7.14.2.86 GSLC\_CB\_PIN\_POLL gslc\_tsGui::pfuncPinPoll**

Callback func ptr for pin polling.

**7.14.2.87 GSLC\_CB\_DRAW gslc\_tsElem::pfuncXDraw**

Callback func ptr for custom drawing.

**7.14.2.88 GSLC\_CB\_EVENT gslc\_tsElem::pfuncXEvent**

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

**7.14.2.89 GSLC\_CB\_TICK gslc\_tsElem::pfuncXTick**

Callback func ptr for timer/main loop tick.

**7.14.2.90 GSLC\_CB\_TOUCH gslc\_tsElem::pfuncXTouch**

Callback func ptr for touch.

**7.14.2.91 const unsigned char\* gslc\_tsImgRef::plmgBuf**

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

**7.14.2.92 char\* gslc\_tsElem::pStrBuf**

Ptr to text string buffer to overlay.

**7.14.2.93 gslc\_tsFont\* gslc\_tsElem::pTxtFont**

Ptr to Font for overlay text.

**7.14.2.94 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.

**7.14.2.95 void\* gslc\_tsGui::pvDriver**

Driver-specific members (gslc\_tsDriver\*)

**7.14.2.96 const void\* gslc\_tsFont::pvFont**

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

**7.14.2.97 void\* gslc\_tsImgRef::pvImgRaw**

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

**7.14.2.98 void\* gslc\_tsEvent::pvScope**

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

**7.14.2.99 void\* gslc\_tsElem::pXData**

Ptr to extended data structure.

**7.14.2.100 uint8\_t gslc\_tsColor::r**

RGB red value.

**7.14.2.101 gslc\_tsRect gslc\_tsPage::rBounds**

Bounding rect for page elements.

**7.14.2.102 gslc\_tsRect gslc\_tsElem::rElem**

Rect region containing element.

**7.14.2.103 gslc\_tsRect gslc\_tsGui::rInvalidateRect**

The rect region that has been invalidated.

**7.14.2.104 gslc\_tsCollect gslc\_tsPage::sCollect**

Collection of elements on page.

**7.14.2.105 gslc\_tsElem gslc\_tsGui::sElemTmpProg**

Temporary element for Flash compatibility.

**7.14.2.106 gslc\_tsImgRef gslc\_tsGui::sImgRefBkgnd**

Image reference for background.

**7.14.2.107 gslc\_tsImgRef gslc\_tsElem::sImgRefGlow**

Image reference to draw (glowing)

**7.14.2.108 gslc\_tsImgRef gslc\_tsElem::sImgRefNorm**

Image reference to draw (normal)

**7.14.2.109 gslc\_tsColor gslc\_tsGui::sTransCol**

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

**7.14.2.110 uint16\_t gslc\_tsRect::w**

Width of region.

**7.14.2.111 int16\_t gslc\_tsRect::x**

X coordinate of corner.

**7.14.2.112 int16\_t gslc\_tsPt::x**

X coordinate.

**7.14.2.113 int16\_t gslc\_tsRect::y**

Y coordinate of corner.

**7.14.2.114 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\\_tsImgRef gslc\\_ResetImage \( \)](#)

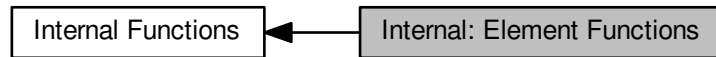
Create a blank image reference structure.

#### 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_teElemRefFlags` 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` sImgRefSel)
 

Set an element to use a bitmap image.
- `bool` `gslc_ElemDrawByRef` (`gslc_tsGui` \*pGui, `gslc_tsElemRef` \*pElemRef, `gslc_teRedrawType` 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_teTxtFlags` 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.

### 7.16.1 Detailed Description

### 7.16.2 Function Documentation

7.16.2.1 `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_tsElemRef* gslc_ElemAdd ( gslc_tsGui * pGui, int16_t nPageld, 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>n↔ Pageld</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_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** `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↔ PageId</i>	ID of page containing element
in	<i>n↔ ElemId</i>	ID of element

**Returns**

none

7.16.2.5 `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 `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_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_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** `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** `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** `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** `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.*
- `int16_t gslc\_PageFocusStep (gslc\_tsGui *pGui, gslc\_tsPage *pPage, bool bNext)`
- `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 `bool gslc_ElemEvent ( void * pvGui, gslc\_tsEvent sEvent )`

Common event handler function for an element.

**Parameters**

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

**Returns**

true if success, false if fail

**7.17.2.2** `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_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 `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_tsPage* gslc_PageFindById ( gslc_tsGui * pGui, int16_t nPageld )`

Find a page in the GUI by its ID.

## Parameters

in	<i>pGui</i>	Pointer to GUI
in	$n \leftrightarrow$ <i>Pageld</i>	Page ID to search

## Returns

Ptr to a page or NULL if none found

7.17.2.6 `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 `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 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 int16\_t `gslc_PageFocusStep ( gslc_tsGui * pGui, gslc_tsPage * pPage, bool bNext )`

**Todo** Doc. This API is experimental and subject to change

7.17.2.10 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

#### 7.17.2.11 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

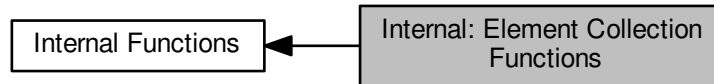
in	<i>pGui</i>	Pointer to GUI
----	-------------	----------------

##### 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_tsElemRefFlags` 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)  
*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)  
*Set the element within a collection that is currently being tracked.*
- `int16_t` `gslc_CollectGetFocus` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Get the element index within a collection that is currently in focus.*
- void `gslc_CollectSetFocus` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `int16_t` nElemInd)  
*Set the element index within a collection that is currently in focus.*
- bool `gslc_CollectFindFocusStep` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, bool bNext, bool \*pbWrapped, `int16_t` \*pnElemInd)  
*Find the next element in focus.*
- 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_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_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_tsElemRef* gslc_CollectFindElemFromCoord ( gslc_tsGui * pGui, gslc_tsCollect * pCollect, int16_t nX, int16_t nY )`

Find an element in a collection by a 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

**Returns**

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

**7.18.2.4** `bool gslc_CollectFindFocusStep ( gslc_tsGui * pGui, gslc_tsCollect * pCollect, bool bNext, bool * pbWrapped, int16_t * pnElemInd )`

**Todo** Doc. This API is experimental and subject to change

**7.18.2.5** `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

**7.18.2.6** `int16_t gslc_CollectGetFocus ( gslc_tsGui * pGui, gslc_tsCollect * pCollect )`

Get the element index within a collection that is currently in focus.

**Parameters**

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

**Returns**

Element index or GSLC\_IND\_NONE for none

7.18.2.7 `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

7.18.2.8 `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

7.18.2.9 `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

7.18.2.10 void `gslc_CollectSetElemTracked` ( `gslc_tsGui` \* *pGui*, `gslc_tsCollect` \* *pCollect*, `gslc_tsElemRef` \* *pElemRef* )

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

#### Returns

none

7.18.2.11 void `gslc_CollectSetFocus` ( `gslc_tsGui` \* *pGui*, `gslc_tsCollect` \* *pCollect*, `int16_t` *nElemInd* )

Set the element index within a collection that is currently in focus.

#### Parameters

in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>nElemInd</i>	Element index to set in focus, <code>GSLC_IND_NONE</code> for none

#### Returns

none

7.18.2.12 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

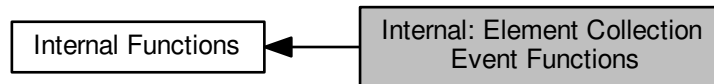
in	<i>pGui</i>	Pointer to GUI
in	<i>pCollect</i>	Pointer to the collection
in	<i>pElemRefParent</i>	Ptr to element reference that is the parent

**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 `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 `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 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 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

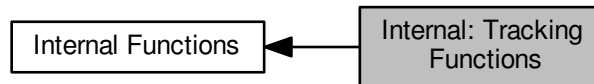
in	<i>pvGui</i>	Void ptr to GUI (typecast to <code>gslc_tsGui*</code> )
in	<i>pvElemRef</i>	Void ptr to Element Reference(typecast to <code>gslc_tsElemRef*</code> )
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element
in	<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_tsPage` \*pPage, `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)

### 7.20.1 Detailed Description

### 7.20.2 Function Documentation

7.20.2.1 bool `gslc_InputMapLookup` ( `gslc_tsGui` \* pGui, `gslc_telInputRawEvent` eInputEvent, `int16_t` nInputVal, `gslc_teAction` \* peAction, `int16_t` \* pnActionVal )

**Todo** Doc. This API is experimental and subject to change

7.20.2.2 void `gslc_TrackInput` ( `gslc_tsGui` \* pGui, `gslc_tsPage` \* pPage, `gslc_telInputRawEvent` 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>pPage</i>	Pointer to current page
in	<i>eInputEvent</i>	Indication of event type
in	<i>nInputVal</i>	Additional data for event type



**Returns**

none

**7.20.2.3** 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**

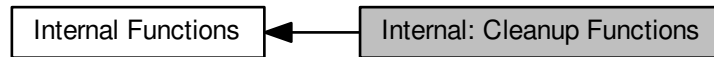
in	<i>pGui</i>	Pointer to GUI
in	<i>pPage</i>	Pointer to current page
in	<i>nX</i>	X coordinate of touch event
in	<i>nY</i>	Y coordinate of touch event
in	<i>nPress</i>	Pressure level of touch event (0 for none, else touch)

**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 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 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 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 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 void gslc\_ResetElem ( gslc\_tsElem \* pElem )**

Initialize an Element struct.

**Parameters**

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

## Returns

none

7.21.2.6 void gslc\_ResetFont ( gslc\_tsFont \* *pFont* )

Initialize a Font struct.

## Parameters

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

## Returns

none



## Chapter 8

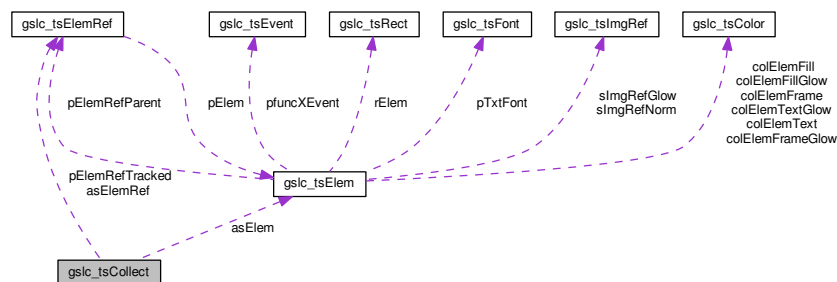
# Data Structure Documentation

### 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 nElemIndFocused`  
*Element index currently in focus (eg. by keyboard/pin control), `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:

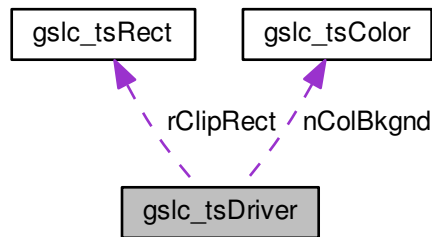
- `src/GUISlice.h`



## 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.*

### 8.3.1 Field Documentation

#### 8.3.1.1 [gslc\\_tsColor](#) [gslc\\_tsDriver::nColBkgnd](#)

Background color (if not image-based)

#### 8.3.1.2 [gslc\\_tsRect](#) [gslc\\_tsDriver::rClipRect](#)

Clipping rectangle.

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

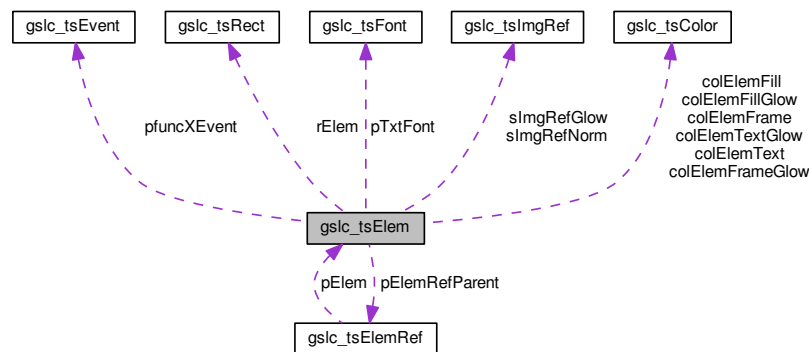
- [src/GUIslice\\_drv\\_adagfx.h](#)
- [src/GUIslice\\_drv\\_m5stack.h](#)
- [src/GUIslice\\_drv\\_tft\\_espi.h](#)
- [src/GUIslice\\_drv\\_utft.h](#)

## 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.*
- `uint8_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:

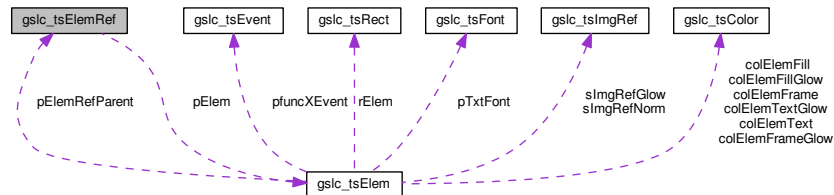
- [src/GUIslice.h](#)

## 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.*

### 8.8.1 Detailed Description

Font reference structure.

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

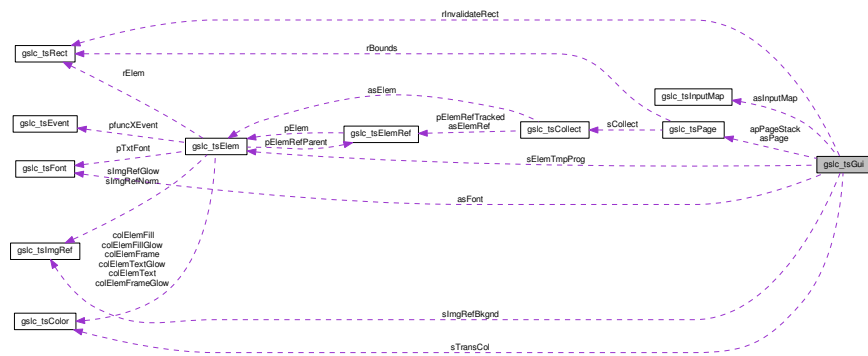
- [src/GUISlice.h](#)

## 8.9 gslc\_tsGui Struct Reference

GUI structure.

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

Collaboration diagram for gslc\_tsGui:



### Data Fields

- `uint16_t nDispW`  
Width of the display (pixels)
- `uint16_t nDispH`  
Height of the display (pixels)
- `uint16_t nDisp0W`  
Width of the display (pixels) in native orientation.
- `uint16_t nDisp0H`  
Height of the display (pixels) in native orientation.
- `uint8_t nDispDepth`  
Bit depth of display (bits per pixel)
- `uint8_t nRotation`  
Adafruit GFX Rotation of display.
- `uint8_t nTouchRotation`  
Touchscreen rotation offset vs display.
- `uint8_t nSwapXY`  
Adafruit GFX Touch Swap x and y axes.
- `uint8_t nFlipX`  
Adafruit GFX Touch Flip x axis.

- [uint8\\_t nFlipY](#)  
*Adafruit GFX Touch Flip x axis.*
- [uint16\\_t nTouchCalXMin](#)  
*Calibration X minimum reading.*
- [uint16\\_t nTouchCalXMax](#)  
*Calibration X maximum reading.*
- [uint16\\_t nTouchCalYMin](#)  
*Calibration Y minimum reading.*
- [uint16\\_t nTouchCalYMax](#)  
*Calibration Y maximum reading.*
- [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 sElemTmpProg](#)  
*Temporary element for Flash compatibility.*
- [gslc\\_telInitStat elInitStatTouch](#)  
*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.*
- [void \\* pvDriver](#)  
*Driver-specific members (gslc\_tsDriver\*)*
- [bool bRedrawPartialEn](#)  
*Driver supports partial page redraw.*
- [gslc\\_tsImgRef slmgRefBkgnd](#)  
*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.*

### 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:

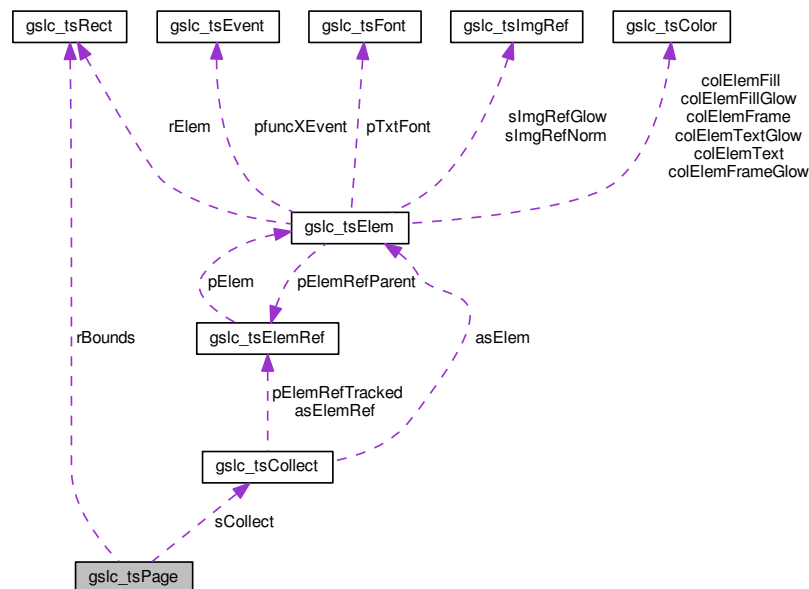
- [src/GUIslice.h](#)

## 8.12 gslc\_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.*

### 8.12.1 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](#)

## 8.13 gslc\_tsPt Struct Reference

Define point coordinates.

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

### Data Fields

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

#### 8.13.1 Detailed Description

Define point coordinates.

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

- `src/GUIslice.h`

## 8.14 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.*

#### 8.14.1 Detailed Description

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

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

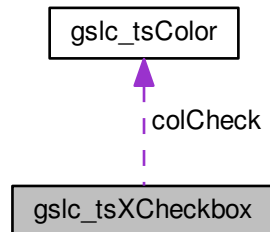
- `src/GUIslice.h`

## 8.15 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.*
- `GSLC_CB_XCHECKBOX` `pfuncXToggle`  
*Callback event to say element has changed.*

### 8.15.1 Detailed Description

Extended data for Checkbox element.

### 8.15.2 Field Documentation

#### 8.15.2.1 bool `gslc_tsXCheckbox::bChecked`

Indicates if it is selected (checked)

#### 8.15.2.2 bool `gslc_tsXCheckbox::bRadio`

Radio-button operation if true.

### 8.15.2.3 gslc\_tsColor gslc\_tsXCheckbox::colCheck

Color of checked inner fill.

### 8.15.2.4 gslc\_teXCheckboxStyle gslc\_tsXCheckbox::nStyle

Drawing style for element.

### 8.15.2.5 GSLC\_CB\_XCHECKBOX gslc\_tsXCheckbox::pfuncXToggle

Callback event to say element has changed.

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

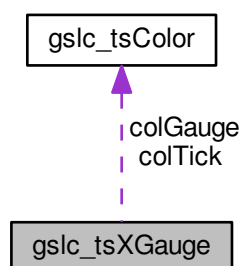
- [src/elem/XCheckbox.h](#)

## 8.16 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.*

### 8.16.1 Detailed Description

Extended data for Gauge element.

### 8.16.2 Field Documentation

#### 8.16.2.1 `bool gslc_tsXGauge::bFlip`

Reverse direction of gauge.

#### 8.16.2.2 `bool gslc_tsXGauge::bIndicFill`

Fill the indicator if true.

#### 8.16.2.3 `bool gslc_tsXGauge::bValLastValid`

Last value valid?

#### 8.16.2.4 `bool gslc_tsXGauge::bVert`

Vertical if true, else Horizontal.

#### 8.16.2.5 `gslc_tsColor gslc_tsXGauge::colGauge`

Color of gauge fill bar.

#### 8.16.2.6 `gslc_tsColor gslc_tsXGauge::colTick`

Color of gauge tick marks.

#### 8.16.2.7 `uint16_t gslc_tsXGauge::nIndicLen`

Indicator length.

#### 8.16.2.8 `uint16_t gslc_tsXGauge::nIndicTip`

Size of tip at end of indicator.

#### 8.16.2.9 `int16_t gslc_tsXGauge::nMax`

Maximum control value.

#### 8.16.2.10 `int16_t gslc_tsXGauge::nMin`

Minimum control value.

#### 8.16.2.11 `gslc_tsXGaugeStyle gslc_tsXGauge::nStyle`

Gauge sub-type.

#### 8.16.2.12 `uint16_t gslc_tsXGauge::nTickCnt`

Number of gauge tick marks.

#### 8.16.2.13 uint16\_t gslc\_tsXGauge::nTickLen

Length of gauge tick marks.

#### 8.16.2.14 int16\_t gslc\_tsXGauge::nVal

Current control value.

#### 8.16.2.15 int16\_t gslc\_tsXGauge::nValLast

Last value.

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

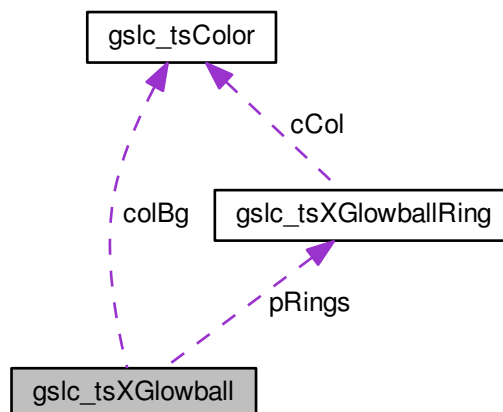
- [src/elem/XGauge.h](#)

## 8.17 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.*

### 8.17.1 Detailed Description

Extended data for Slider element.

### 8.17.2 Field Documentation

#### 8.17.2.1 `gslc_tsColor gslc_tsXGlowball::colBg`

Background color (for redraw)

#### 8.17.2.2 `int16_t gslc_tsXGlowball::nAngEnd`

Ending angle (0..510 degrees)

#### 8.17.2.3 `int16_t gslc_tsXGlowball::nAngStart`

Starting angle (0..510 degrees)

#### 8.17.2.4 `int16_t gslc_tsXGlowball::nMidX`

Gauge midpoint X coord.

#### 8.17.2.5 `int16_t gslc_tsXGlowball::nMidY`

Gauge midpoint Y coord.

#### 8.17.2.6 `uint8_t gslc_tsXGlowball::nNumRings`

Number of rings in definition.

#### 8.17.2.7 `uint16_t gslc_tsXGlowball::nQuality`

Rendering quality (number of segments / rotation)

#### 8.17.2.8 `int16_t gslc_tsXGlowball::nVal`

Current value.

#### 8.17.2.9 `int16_t gslc_tsXGlowball::nValLast`

Previous value.

#### 8.17.2.10 `gslc_tsXGlowballRing* gslc_tsXGlowball::pRings`

Ring definition array.

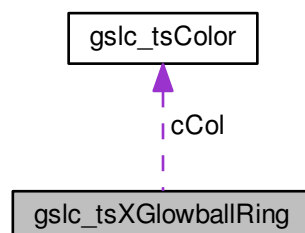
The documentation for this struct was generated from the following file:

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

## 8.18 `gslc_tsXGlowballRing` Struct Reference

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

Collaboration diagram for `gslc_tsXGlowballRing`:



## Data Fields

- [uint8\\_t nRad1](#)
- [uint8\\_t nRad2](#)
- [gslc\\_tsColor cCol](#)

### 8.18.1 Field Documentation

8.18.1.1 [gslc\\_tsColor](#) [gslc\\_tsXGlowballRing::cCol](#)

8.18.1.2 [uint8\\_t](#) [gslc\\_tsXGlowballRing::nRad1](#)

8.18.1.3 [uint8\\_t](#) [gslc\\_tsXGlowballRing::nRad2](#)

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

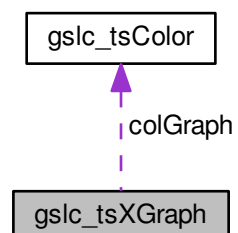
- [src/elem/XGlowball.h](#)

## 8.19 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.*

### 8.19.1 Detailed Description

Extended data for Graph element.

### 8.19.2 Field Documentation

#### 8.19.2.1 `bool gslc_tsXGraph::bScrollEn`

Enable for scrollbar.

#### 8.19.2.2 `gslc_tsColor gslc_tsXGraph::colGraph`

Color of the graph.

#### 8.19.2.3 `gslc_tsXGraphStyle` `gslc_tsXGraph::eStyle`

Style of the graph.

#### 8.19.2.4 `uint16_t` `gslc_tsXGraph::nBufCnt`

Number of points in buffer.

#### 8.19.2.5 `uint16_t` `gslc_tsXGraph::nBufMax`

Maximum number of points in buffer.

#### 8.19.2.6 `uint8_t` `gslc_tsXGraph::nMargin`

Margin for graph area within element rect.

#### 8.19.2.7 `uint16_t` `gslc_tsXGraph::nPlotIndMax`

Number of data points to show in window.

#### 8.19.2.8 `uint16_t` `gslc_tsXGraph::nPlotIndStart`

First row of current window.

#### 8.19.2.9 `int16_t` `gslc_tsXGraph::nPlotValMax`

Visible window maximum value.

#### 8.19.2.10 `int16_t` `gslc_tsXGraph::nPlotValMin`

Visible window minimum value.

#### 8.19.2.11 `uint16_t` `gslc_tsXGraph::nScrollPos`

Current scrollbar position.

#### 8.19.2.12 `uint16_t` `gslc_tsXGraph::nWndHeight`

Visible window height.

### 8.19.2.13 uint16\_t gslc\_tsXGraph::nWndWidth

Visible window width.

### 8.19.2.14 int16\_t\* gslc\_tsXGraph::pBuf

Ptr to the data buffer (circular buffer)

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

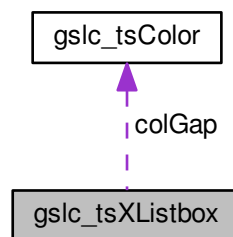
- [src/elem/XGraph.h](#)

## 8.20 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 [bItemAutoSizeW](#)  
*Enable auto-sizing of items (in width)*
- bool [bItemAutoSizeH](#)  
*Enable auto-sizing of items (in height)*
- int16\_t [nItemCurSel](#)  
*Currently selected item (XLISTBOX\_SEL\_NONE for none)*
- int16\_t [nItemCurSelLast](#)  
*Old selected item to redraw (XLISTBOX\_SEL\_NONE for none)*
- int16\_t [nItemSavedSel](#)  
*Persistent selected item (ie. saved selection)*
- int16\_t [nItemTop](#)  
*Item to show at top of list after scrolling (0 is default)*
- [GSLC\\_CB\\_XLISTBOX\\_SEL](#) [pfuncXSel](#)  
*Callback func ptr for selection update.*

### 8.20.1 Detailed Description

Extended data for Listbox element.

### 8.20.2 Field Documentation

#### 8.20.2.1 bool `gslc_tsXListbox::bItemAutoSizeH`

Enable auto-sizing of items (in height)

#### 8.20.2.2 bool `gslc_tsXListbox::bItemAutoSizeW`

Enable auto-sizing of items (in width)

#### 8.20.2.3 bool `gslc_tsXListbox::bNeedRecalc`

Determine if sizing may need recalc.

#### 8.20.2.4 `gslc_tsColor gslc_tsXListbox::colGap`

Gap color.

#### 8.20.2.5 `uint16_t gslc_tsXListbox::nBufItemsMax`

Max size of buffer containing items.

#### 8.20.2.6 `uint16_t gslc_tsXListbox::nBufItemsPos`

Current buffer position.

#### 8.20.2.7 `int8_t gslc_tsXListbox::nCols`

Number of columns.

#### 8.20.2.8 `int16_t gslc_tsXListbox::nItemCnt`

Number of items in the list.

#### 8.20.2.9 `int16_t gslc_tsXListbox::nItemCurSel`

Currently selected item (XLISTBOX\_SEL\_NONE for none)

#### 8.20.2.10 `int16_t gslc_tsXListbox::nItemCurSelLast`

Old selected item to redraw (XLISTBOX\_SEL\_NONE for none)

#### 8.20.2.11 `int8_t gslc_tsXListbox::nItemGap`

Gap between listbox items.

#### 8.20.2.12 `int16_t gslc_tsXListbox::nItemH`

Height of listbox item.

#### 8.20.2.13 `int16_t gslc_tsXListbox::nItemSavedSel`

Persistent selected item (ie. saved selection)



## 8.20.2.14 int16\_t gslc\_tsXListBox::nItemTop

Item to show at top of list after scrolling (0 is default)

## 8.20.2.15 int16\_t gslc\_tsXListBox::nItemW

Width of listbox item.

## 8.20.2.16 int8\_t gslc\_tsXListBox::nMarginH

Margin inside main listbox area (Y offset)

## 8.20.2.17 int8\_t gslc\_tsXListBox::nMarginW

Margin inside main listbox area (X offset)

## 8.20.2.18 int8\_t gslc\_tsXListBox::nRows

Number of columns (or XLSITBOX\_SIZE\_AUTO to calculate)

## 8.20.2.19 uint8\_t\* gslc\_tsXListBox::pBufItems

Buffer containing items.

## 8.20.2.20 GSLC\_CB\_XLISTBOX\_SEL gslc\_tsXListBox::pfuncXSel

Callback func ptr for selection update.

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

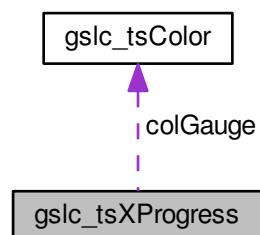
- [src/elem/XListBox.h](#)

## 8.21 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.*

### 8.21.1 Detailed Description

Extended data for Gauge element.

### 8.21.2 Field Documentation

#### 8.21.2.1 `bool gslc_tsXProgress::bFlip`

Reverse direction of gauge.

#### 8.21.2.2 `bool gslc_tsXProgress::bValLastValid`

Last value valid?

#### 8.21.2.3 `bool gslc_tsXProgress::bVert`

Vertical if true, else Horizontal.

#### 8.21.2.4 `gslc_tsColor gslc_tsXProgress::colGauge`

Color of gauge fill bar.

#### 8.21.2.5 `int16_t gslc_tsXProgress::nMax`

Maximum control value.

## 8.21.2.6 int16\_t gslc\_tsXProgress::nMin

Minimum control value.

## 8.21.2.7 int16\_t gslc\_tsXProgress::nVal

Current control value.

## 8.21.2.8 int16\_t gslc\_tsXProgress::nValLast

Last value.

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

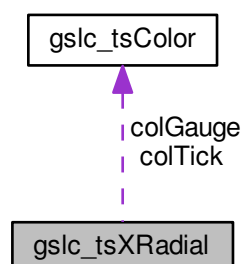
- [src/elem/XProgress.h](#)

## 8.22 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.*

### 8.22.1 Detailed Description

Extended data for Gauge element.

### 8.22.2 Field Documentation

#### 8.22.2.1 `bool gslc_tsXRadial::bFlip`

Reverse direction of gauge.

#### 8.22.2.2 `bool gslc_tsXRadial::bIndicFill`

Fill the indicator if true.

#### 8.22.2.3 `bool gslc_tsXRadial::bValLastValid`

Last value valid?

**8.22.2.4 gslc\_tsColor gslc\_tsXRadial::colGauge**

Color of gauge fill bar.

**8.22.2.5 gslc\_tsColor gslc\_tsXRadial::colTick**

Color of gauge tick marks.

**8.22.2.6 uint16\_t gslc\_tsXRadial::nIndicLen**

Indicator length.

**8.22.2.7 uint16\_t gslc\_tsXRadial::nIndicTip**

Size of tip at end of indicator.

**8.22.2.8 int16\_t gslc\_tsXRadial::nMax**

Maximum control value.

**8.22.2.9 int16\_t gslc\_tsXRadial::nMin**

Minimum control value.

**8.22.2.10 uint16\_t gslc\_tsXRadial::nTickCnt**

Number of gauge tick marks.

**8.22.2.11 uint16\_t gslc\_tsXRadial::nTickLen**

Length of gauge tick marks.

**8.22.2.12 int16\_t gslc\_tsXRadial::nVal**

Current control value.

**8.22.2.13 int16\_t gslc\_tsXRadial::nValLast**

Last value.

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

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

## 8.23 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?*

### 8.23.1 Detailed Description

Extended data for Gauge element.

### 8.23.2 Field Documentation

#### 8.23.2.1 `bool gslc_tsXRamp::bValLastValid`

Last value valid?

#### 8.23.2.2 `int16_t gslc_tsXRamp::nMax`

Maximum control value.

#### 8.23.2.3 `int16_t gslc_tsXRamp::nMin`

Minimum control value.

#### 8.23.2.4 `int16_t gslc_tsXRamp::nVal`

Current control value.

## 8.23.2.5 int16\_t gslc\_tsXRamp::nValLast

Last value.

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

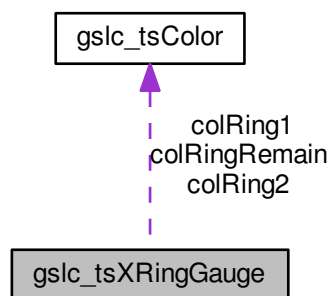
- [src/elem/XRamp.h](#)

## 8.24 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](#)]

### 8.24.1 Detailed Description

Extended data for XRingGauge element.

### 8.24.2 Field Documentation

8.24.2.1 `char gslc_tsXRingGauge::acStrLast[XRING_STR_MAX]`

8.24.2.2 `bool gslc_tsXRingGauge::bGradient`

8.24.2.3 `gslc_tsColor gslc_tsXRingGauge::colRing1`

8.24.2.4 `gslc_tsColor gslc_tsXRingGauge::colRing2`

8.24.2.5 `gslc_tsColor gslc_tsXRingGauge::colRingRemain`

8.24.2.6 `int16_t gslc_tsXRingGauge::nAngRange`

8.24.2.7 `int16_t gslc_tsXRingGauge::nAngStart`

8.24.2.8 `int16_t gslc_tsXRingGauge::nQuality`

8.24.2.9 `uint8_t gslc_tsXRingGauge::nSegGap`

8.24.2.10 `int8_t gslc_tsXRingGauge::nThickness`

8.24.2.11 `int16_t gslc_tsXRingGauge::nVal`

Current position value.

8.24.2.12 `int16_t gslc_tsXRingGauge::nValLast`

Previous position value.

8.24.2.13 `int16_t gslc_tsXRingGauge::nValMax`

8.24.2.14 `int16_t gslc_tsXRingGauge::nValMin`

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

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

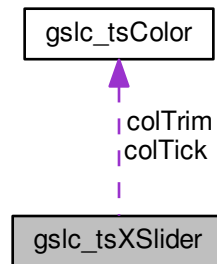


## 8.25 gslc\_tsXSlider Struct Reference

Extended data for Slider element.

```
#include <XSlider.h>
```

Collaboration diagram for gslc\_tsXSlider:



### Data Fields

- bool `bVert`  
*Orientation: true if vertical, else horizontal.*
- 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.*

### 8.25.1 Detailed Description

Extended data for Slider element.

## 8.25.2 Field Documentation

### 8.25.2.1 `bool gslc_tsXSlider::bTrim`

Style: show a trim color.

### 8.25.2.2 `bool gslc_tsXSlider::bVert`

Orientation: true if vertical, else horizontal.

### 8.25.2.3 `gslc_tsColor gslc_tsXSlider::colTick`

Style: color of ticks.

### 8.25.2.4 `gslc_tsColor gslc_tsXSlider::colTrim`

Style: color of trim.

### 8.25.2.5 `int16_t gslc_tsXSlider::nPos`

Current position value of the slider.

### 8.25.2.6 `int16_t gslc_tsXSlider::nPosMax`

Maximum position value of the slider.

### 8.25.2.7 `int16_t gslc_tsXSlider::nPosMin`

Minimum position value of the slider.

### 8.25.2.8 `int16_t gslc_tsXSlider::nThumbSz`

Size of the thumb control.

### 8.25.2.9 `uint16_t gslc_tsXSlider::nTickDiv`

Style: number of tickmark divisions (0 for none)

### 8.25.2.10 `int16_t gslc_tsXSlider::nTickLen`

Style: length of tickmarks.

#### 8.25.2.11 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](#)

## 8.26 gslc\_tsXTemplate Struct Reference

Callback function for slider feedback.

```
#include <XTemplate.h>
```

### 8.26.1 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](#)

## 8.27 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)*

### 8.27.1 Detailed Description

Extended data for Textbox element.

### 8.27.2 Field Documentation

#### 8.27.2.1 bool gslc\_tsXTextbox::bScrollEn

Enable for scrollbar.

#### 8.27.2.2 bool gslc\_tsXTextbox::bWrapEn

Enable for line wrapping.

#### 8.27.2.3 uint16\_t gslc\_tsXTextbox::nBufCols

Number of columns in buffer.

#### 8.27.2.4 uint8\_t gslc\_tsXTextbox::nBufPosX

Buffer X position.

**8.27.2.5    uint8\_t gslc\_tsXTextbox::nBufPosY**

Buffer Y position.

**8.27.2.6    uint16\_t gslc\_tsXTextbox::nBufRows**

Number of rows in buffer.

**8.27.2.7    uint8\_t gslc\_tsXTextbox::nChSizeX**

Width of characters (pixels)

**8.27.2.8    uint8\_t gslc\_tsXTextbox::nChSizeY**

Height of characters (pixels)

**8.27.2.9    uint8\_t gslc\_tsXTextbox::nCurPosX**

Cursor X position.

**8.27.2.10    uint8\_t gslc\_tsXTextbox::nCurPosY**

Cursor Y position.

**8.27.2.11    int8\_t gslc\_tsXTextbox::nMarginX**

Margin for text area within element rect (X)

**8.27.2.12    int8\_t gslc\_tsXTextbox::nMarginY**

Margin for text area within element rect (Y)

**8.27.2.13    int16\_t gslc\_tsXTextbox::nRedrawRow**

Specific row to update in redraw (if not -1)

**8.27.2.14    uint16\_t gslc\_tsXTextbox::nScrollPos**

Current scrollbar position.

#### 8.27.2.15 `uint8_t gslc_tsXTextbox::nWndCols`

Window X size.

#### 8.27.2.16 `uint8_t gslc_tsXTextbox::nWndRows`

Window Y size.

#### 8.27.2.17 `uint8_t gslc_tsXTextbox::nWndRowStart`

First row of current window.

#### 8.27.2.18 `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](#)

## 8.28 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](#)

### 8.28.1 Constructor & Destructor Documentation

8.28.1.1 THPoint::THPoint ( void )

8.28.1.2 THPoint::THPoint ( uint16\_t x, uint16\_t y, uint16\_t z )

### 8.28.2 Member Function Documentation

8.28.2.1 bool THPoint::operator!= ( THPoint *p1* )

8.28.2.2 bool THPoint::operator== ( THPoint *p1* )

### 8.28.3 Field Documentation

8.28.3.1 uint16\_t THPoint::x

8.28.3.2 uint16\_t THPoint::y

8.28.3.3 uint16\_t THPoint::z

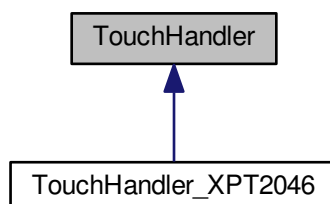
The documentation for this class was generated from the following files:

- [src/GUISlice\\_th.h](#)
- [src/GUISlice\\_th.cpp](#)

## 8.29 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)

### 8.29.1 Constructor & Destructor Documentation

8.29.1.1 [TouchHandler::TouchHandler](#) ( ) `[inline]`

### 8.29.2 Member Function Documentation

8.29.2.1 void [TouchHandler::begin](#) ( void ) `[virtual]`

Reimplemented in [TouchHandler\\_XPT2046](#).

8.29.2.2 [THPoint](#) [TouchHandler::getPoint](#) ( void ) `[virtual]`

Reimplemented in [TouchHandler\\_XPT2046](#).

8.29.2.3 [THPoint](#) [TouchHandler::scale](#) ( [THPoint](#) pIn )

8.29.2.4 void [TouchHandler::setCalibration](#) ( uint16\_t ts\_xMin, uint16\_t ts\_xMax, uint16\_t ts\_yMin, uint16\_t ts\_yMax )

8.29.2.5 void [TouchHandler::setSize](#) ( uint16\_t \_disp\_xSize, uint16\_t \_disp\_ySize )

8.29.2.6 void [TouchHandler::setSwapFlip](#) ( bool \_swapXY, bool \_flipX, bool \_flipY )

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

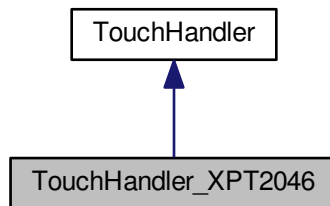
- src/[GUIslice\\_th.h](#)
- src/[GUIslice\\_th.cpp](#)



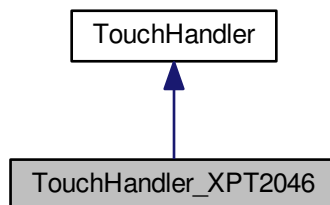
## 8.30 TouchHandler\_XPT2046 Class Reference

```
#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](#)

### 8.30.1 Constructor & Destructor Documentation

8.30.1.1 `TouchHandler_XPT2046::TouchHandler_XPT2046 ( SPIClass & spi, uint8_t spi_cs_pin )` `[inline]`

### 8.30.2 Member Function Documentation

8.30.2.1 `void TouchHandler_XPT2046::begin ( void )` `[inline],[virtual]`

Reimplemented from [TouchHandler](#).

8.30.2.2 `THPoint TouchHandler_XPT2046::getPoint ( void )` `[inline],[virtual]`

Reimplemented from [TouchHandler](#).

### 8.30.3 Field Documentation

8.30.3.1 `SPIClass TouchHandler_XPT2046::spi`

8.30.3.2 `XPT2046_touch TouchHandler_XPT2046::touchDriver`

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

- [src/GUIslice\\_th\\_XPT2046.h](#)

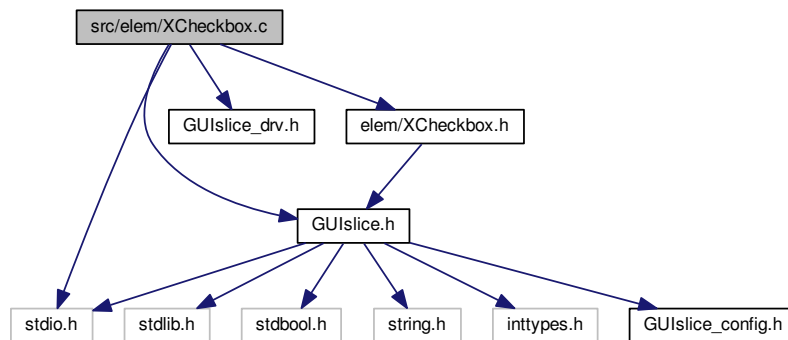
## Chapter 9

# 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)
- 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\\_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 [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 <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

### 9.2.1.3 `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 `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 `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** 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** void gslc\_ElemXCheckboxSetStateHelp ( gslc\_tsGui \* *pGui*, gslc\_tsElemRef \* *pElemRef*, bool *bChecked* )

**9.2.1.8** 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** 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 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

### 9.2.2 Variable Documentation

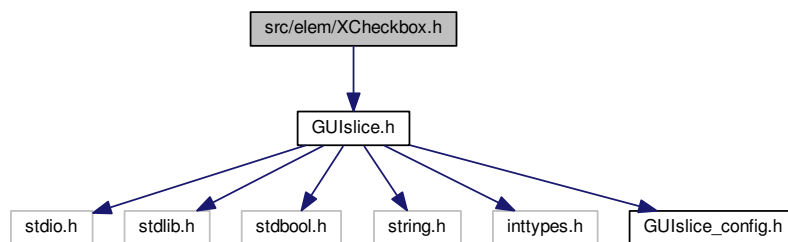
9.2.2.1 `const char ERRSTR_NULL`

9.2.2.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

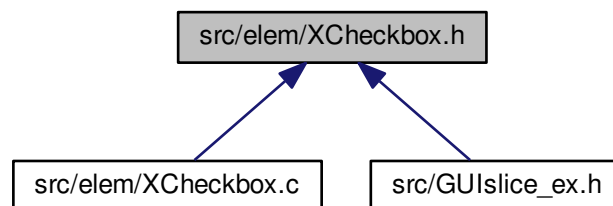
## 9.3 src/elem/XCheckbox.h File Reference

```
#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 `#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
in	<i>col↔ Check_</i>	Color for inner fill when checked
in	<i>b↔ Checked↔ _</i>	Default state

## Returns

none

## 9.3.1.2 #define GSLC\_TYPEX\_CHECKBOX

## 9.3.2 Typedef Documentation

## 9.3.2.1 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 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_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 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\_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 `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 `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 `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 `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** `bool gslc_ElemXCheckboxTouch ( void * pvGui, void * pElemRef, 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 gslc_tsGui*)
in	<i>pElemRef</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

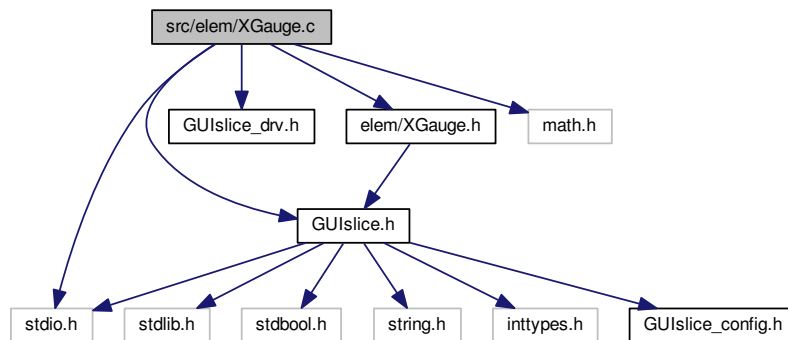
## Returns

true if success, false otherwise

## 9.4 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\\_tsXGaugeStyle](#) 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_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 `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** `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** `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** `void gslc_ElemXGaugeSetIndicator ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, gslc_tsColor colGauge,  
uint16_t nIndicLen, uint16_t nIndicTip, bool blndicFill )`

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** 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** 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 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**

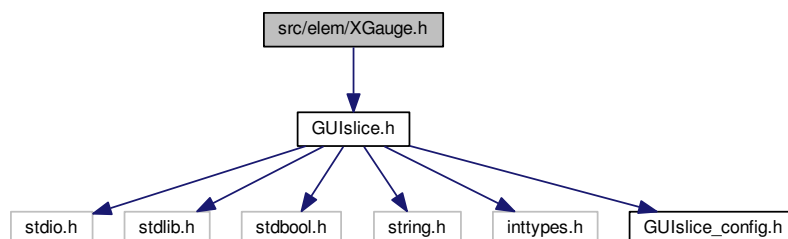
9.4.2.1 const char GSLC\_PMEM\_ERRSTR\_NULL[]

9.4.2.2 const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL[]

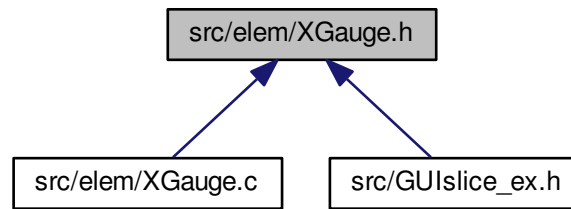
**9.5 src/elem/XGauge.h File Reference**

```
#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\_, col←  
Frame\_, 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\\_tsX←  
Gauge](#) \*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 `#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 `#define GSLC_TYPEX_GAUGE`

### 9.5.2 Enumeration Type Documentation

9.5.2.1 `enum gslc_teXGaugeStyle`

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_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

**9.5.3.2** `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

**9.5.3.3** `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** `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** `void gslc_ElemXGaugeSetIndicator ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, gslc_tsColor colGauge,  
uint16_t nIndicLen, uint16_t nIndicTip, bool blndicFill )`

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** 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** 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** 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

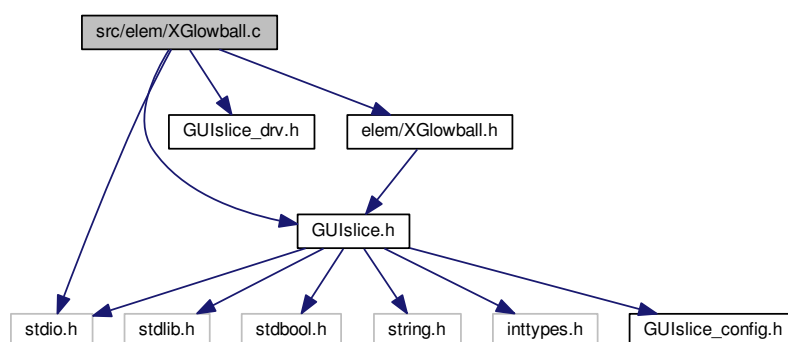
**Returns**

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** 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** 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** 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\\_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 `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

9.6.1.6 `void gslc_ElemXGlowballSetAngles ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, int16_t nAngStart, int16_t nAngEnd )`

9.6.1.7 `void gslc_ElemXGlowballSetColorBack ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, gslc_tsColor colBg )`

9.6.1.8 `void gslc_ElemXGlowballSetQuality ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, uint16_t nQuality )`

9.6.1.9 `void gslc_ElemXGlowballSetVal ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef, int16_t nVal )`

**9.6.2 Variable Documentation**

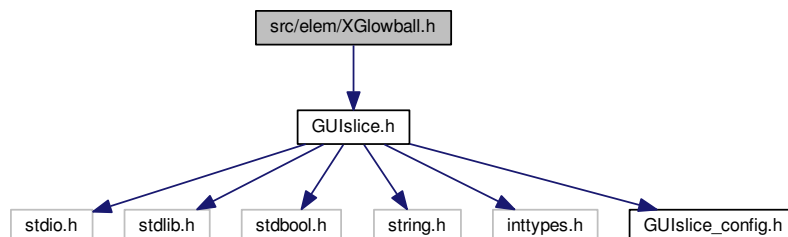
9.6.2.1 `const char GSLC_PMEM_ERRSTR_NULL[]`

9.6.2.2 `const char GSLC_PMEM_ERRSTR_PXD_NULL[]`

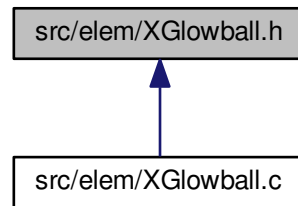
**9.7 src/elem/XGlowball.h File Reference**

```
#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 #define GSLC\_TYPEX\_GLOW

## 9.7.2 Function Documentation

9.7.2.1 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 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 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\_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 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

9.7.2.6 void `gslc_ElemXGlowballSetAngles` ( `gslc_tsGui` \* *pGui*, `gslc_tsElemRef` \* *pElemRef*, `int16_t` *nAngStart*, `int16_t` *nAngEnd* )

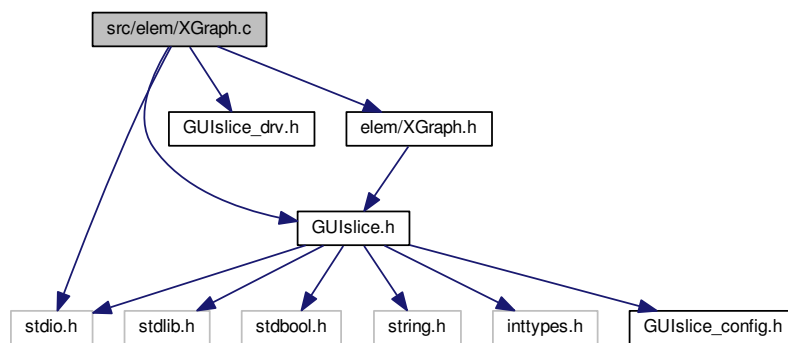
9.7.2.7 void `gslc_ElemXGlowballSetColorBack` ( `gslc_tsGui` \* *pGui*, `gslc_tsElemRef` \* *pElemRef*, `gslc_tsColor` *colBg* )

9.7.2.8 void `gslc_ElemXGlowballSetQuality` ( `gslc_tsGui` \* *pGui*, `gslc_tsElemRef` \* *pElemRef*, `uint16_t` *nQuality* )

9.7.2.9 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.*
- bool `gslc_ElemXGraphDraw` ( void \* *pGui*, void \* *pElemRef*, `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** 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\\_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 <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>nFontId</i>	Font ID to use for graph area
in	<i>pBuf</i>	Ptr to data buffer (already allocated) with size ( <a href="#">nBufMax</a> ) 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** bool [gslc\\_ElemXGraphDraw](#) ( void \* *pGui*, void \* *pElemRef*, [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 <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

**9.8.1.4** 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

**9.8.1.5** 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

**9.8.1.6** 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

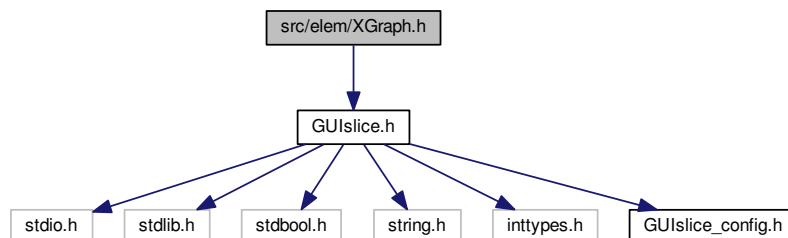
## 9.8.2 Variable Documentation

9.8.2.1 `const char GSLC_PMEM ERRSTR_NULL[]`9.8.2.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

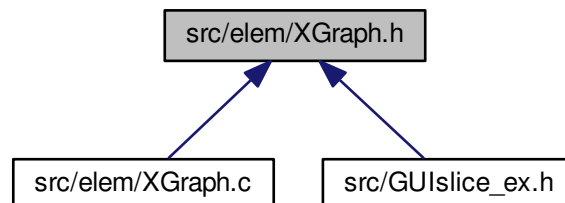
## 9.9 src/elem/XGraph.h File Reference

#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 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.*
- `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_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 `#define GSLC_TYPEX_GRAPH`

### 9.9.2 Enumeration Type Documentation

#### 9.9.2.1 enum `gslc_teXGraphStyle`

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 `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_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** `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** 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

**9.9.3.5** 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

**9.9.3.6** 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

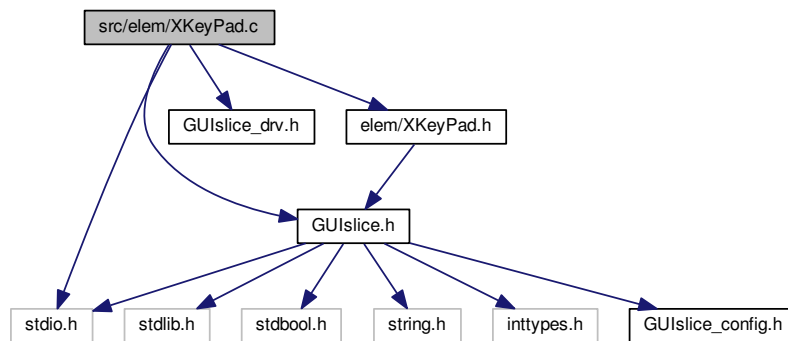
## Returns

none

## 9.10 src/elem/XKeyPad.c File Reference

```
#include "GUIslice.h"  
#include "GUIslice_drv.h"  
#include "elem/XKeyPad.h"  
#include <stdio.h>
```

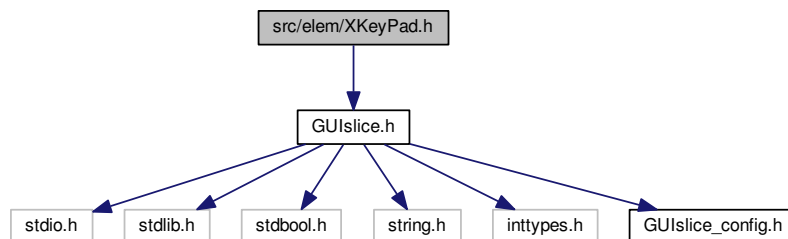
Include dependency graph for XKeyPad.c:



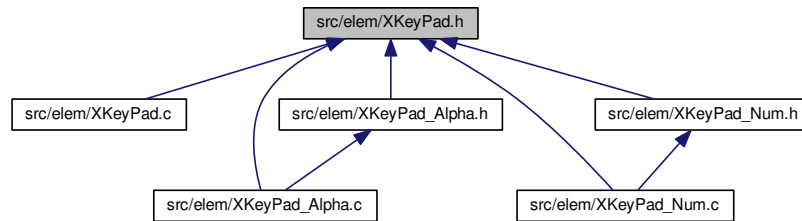
## 9.11 src/elem/XKeyPad.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XKeyPad.h:



This graph shows which files directly or indirectly include this file:



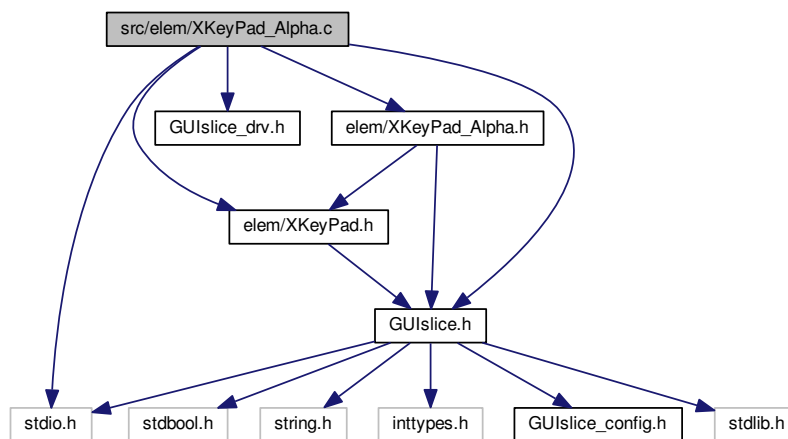
## 9.12 src/elem/XKeyPad\_Alpha.c File Reference

```

#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Alpha.h"
#include <stdio.h>

```

Include dependency graph for XKeyPad\_Alpha.c:



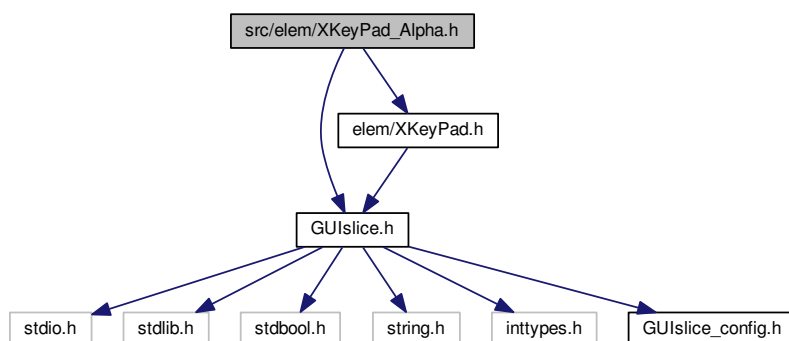
## 9.13 src/elem/XKeyPad\_Alpha.h File Reference

```

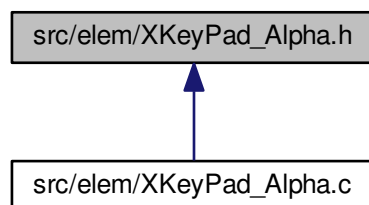
#include "GUIslice.h"
#include "elem/XKeyPad.h"

```

Include dependency graph for XKeyPad\_Alpha.h:



This graph shows which files directly or indirectly include this file:



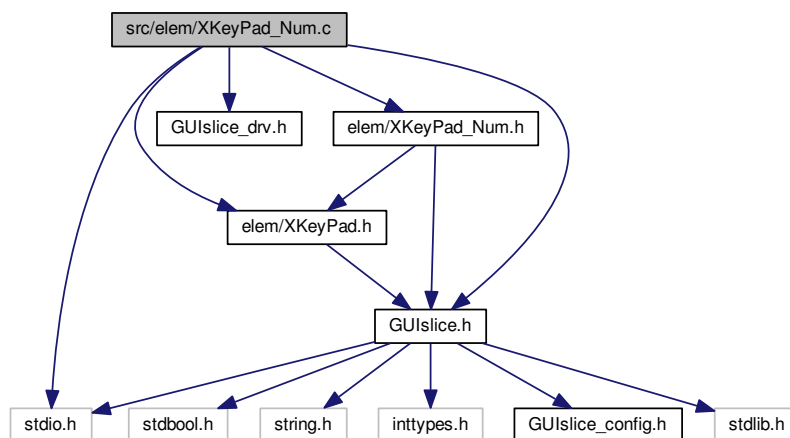
## 9.14 src/elem/XKeyPad\_Num.c File Reference

```

#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XKeyPad.h"
#include "elem/XKeyPad_Num.h"
#include <stdio.h>

```

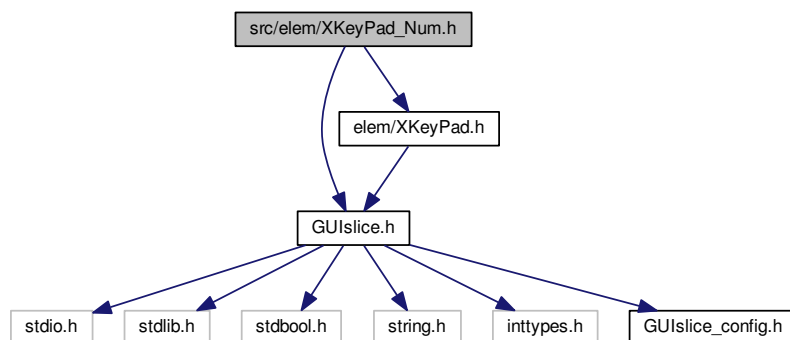
Include dependency graph for XKeyPad\_Num.c:



## 9.15 src/elem/XKeyPad\_Num.h File Reference

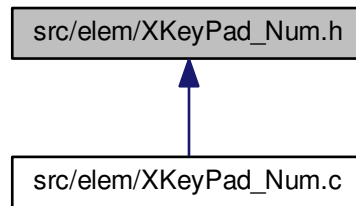
```
#include "GUIslice.h"
#include "elem/XKeyPad.h"
```

Include dependency graph for XKeyPad\_Num.h:





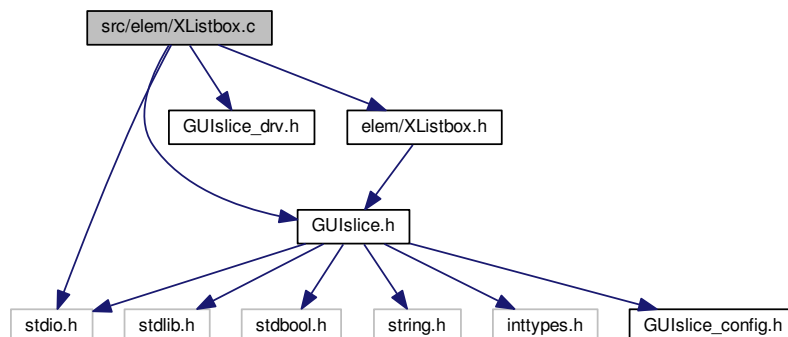
This graph shows which files directly or indirectly include this file:



## 9.16 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` [XLSTBOX\\_MAX\\_STR](#)

### Functions

- `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\\_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](#) []

## 9.16.1 Macro Definition Documentation

### 9.16.1.1 #define XLISTBOX\_MAX\_STR

## 9.16.2 Function Documentation

### 9.16.2.1 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)

**9.16.2.2** `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

**9.16.2.3** `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 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

**9.16.2.4** `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)

**9.16.2.5** `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

**9.16.2.6** `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)

**9.16.2.7** 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

**9.16.2.8** 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

**9.16.2.9** bool gslc\_ElemXListboxRecalcSize ( gslc\_tsXListbox \* *pListbox*, gslc\_tsRect *rElem* )

**9.16.2.10** 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

9.16.2.11 void gslc\_ElemXListboxSetMargin ( gslc\_tsGui \* *pGui*, gslc\_tsElemRef \* *pElemRef*, int8\_t *nMarginW*, int8\_t *nMarginH* )

Configure the margin inside the listbox.

- Defines the region between 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

9.16.2.12 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 <i>nScrollPos</i> item is at the top (0 default)

#### Returns

true if success, false if fail

9.16.2.13 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

9.16.2.14 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

9.16.2.15 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 <code>XLISTBOX_SIZE_AUTO</code> to base on content)
in	<i>nCols</i>	Number of columns ( $\geq 1$ )

#### Returns

none

9.16.2.16 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 ref (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

### 9.16.3 Variable Documentation

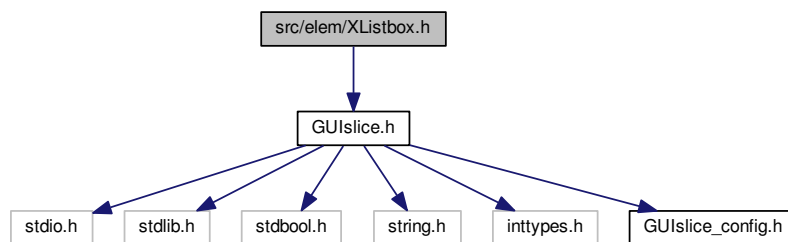
9.16.3.1 `const char GSLC_PMEM ERRSTR_NULL[]`

9.16.3.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

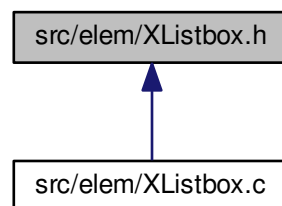
## 9.17 `src/elem/XListbox.h` File Reference

```
#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\\_tsElemRef](#) \*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\\_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.*

## 9.17.1 Macro Definition Documentation

9.17.1.1 `#define GSLC_TYPEX_LISTBOX`

9.17.1.2 `#define XLISTBOX_BUF_OH_R`

9.17.1.3 `#define XLISTBOX_SEL_NONE`

9.17.1.4 `#define XLISTBOX_SIZE_AUTO`

## 9.17.2 Typedef Documentation

9.17.2.1 `typedef bool(* GSLC_CB_XLISTBOX_SEL)(void *pvGui, void *pvElem, int16_t nSel)`

Callback function for Listbox feedback.

## 9.17.3 Function Documentation

9.17.3.1 `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)

9.17.3.2 `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

## Parameters

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

### 9.17.3.3 bool gslc\_ElemXListboxDraw ( void \* *pGui*, void \* *pElemRef*, gslc\_teRedrawType *eRedraw* )

Draw a Listbox element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

## Parameters

in	<i>pGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pElemRef</i>	Void ptr to Element (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

### 9.17.3.4 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)

9.17.3.5 `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

9.17.3.6 `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)

9.17.3.7 `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

9.17.3.8 `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

**9.17.3.9** 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

**9.17.3.10** 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

**9.17.3.11** 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 <i>nScrollPos</i> item is at the top (0 default)

## Returns

true if success, false if fail

**9.17.3.12** `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

**9.17.3.13** `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

**9.17.3.14** `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

9.17.3.15 `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 ref (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**

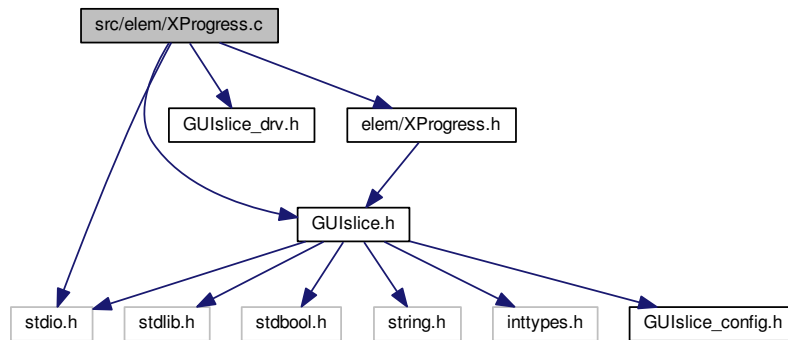
true if success, false otherwise

## 9.18 `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.*
- `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 []`

### 9.18.1 Function Documentation

- 9.18.1.1 `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

**9.18.1.2** `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 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

**9.18.1.3** `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

**9.18.1.4** 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

**9.18.1.5** 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

**9.18.2 Variable Documentation**

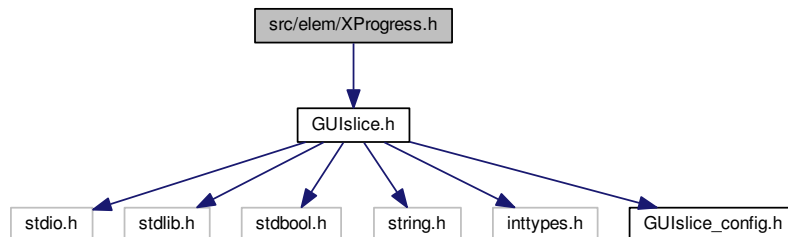
**9.18.2.1** const char GSLC\_PMEM\_ERRSTR\_NULL[]

**9.18.2.2** const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL[]

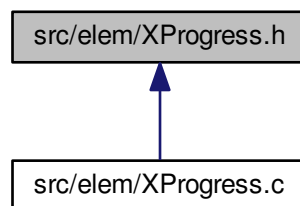
## 9.19 src/elem/XProgress.h File Reference

```
#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.*
- `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.*

### 9.19.1 Macro Definition Documentation

9.19.1.1 `#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

9.19.1.2 `#define GSLC_TYPEX_PROGRESS`

## 9.19.2 Function Documentation

**9.19.2.1** `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 <i>nVal</i> comparison
in	<i>nMax</i>	Maximum value of gauge for <i>nVal</i> 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

**9.19.2.2** `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

9.19.2.3 `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

9.19.2.4 `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

9.19.2.5 `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

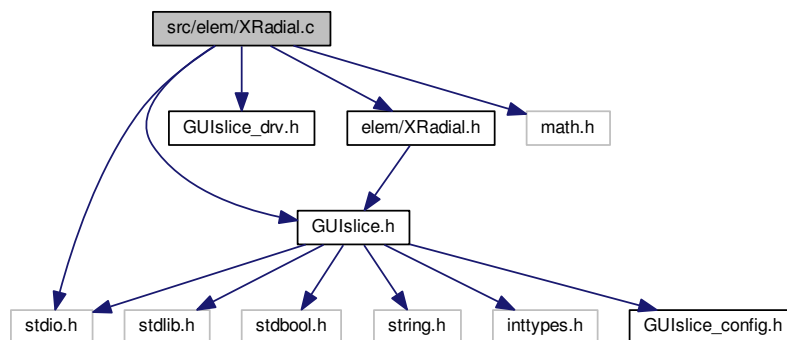
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

## Returns

none

## 9.20 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](#) []

## 9.20.1 Function Documentation

**9.20.1.1** [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

**9.20.1.2** 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

**9.20.1.3** `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

**9.20.1.4** `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 )`

**9.20.1.5** `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

**9.20.1.6** void gslc\_ElemXRadialSetIndicator ( gslc\_tsGui \* *pGui*, gslc\_tsElemRef \* *pElemRef*, gslc\_tsColor *colGauge*, uint16\_t *nIndicLen*, uint16\_t *nIndicTip*, bool *blndicFill* )

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>blndicFill</i>	Fill in the indicator if true

**Returns**

none

**9.20.1.7** 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

**9.20.1.8** 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

**9.20.2 Variable Documentation**

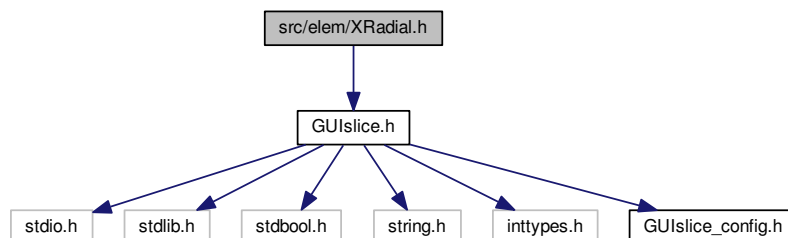
9.20.2.1 `const char GSLC_PMEM ERRSTR_NULL[]`

9.20.2.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

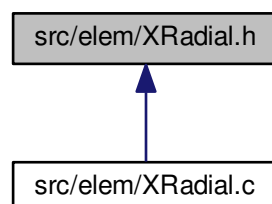
**9.21 src/elem/XRadial.h File Reference**

```
#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\_, colFrame\_, colFill\_, colGauge\_)  
*Create a Gauge Element in Flash.*

## 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.*
- bool [gslc\\_ElemXRadialDrawRadial](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_teRedrawType](#) eRedraw)  
*Helper function to draw a gauge with style: radial.*

### 9.21.1 Macro Definition Documentation

- 9.21.1.1 #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

**Parameters**

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

**9.21.1.2 #define GSLC\_TYPEX\_RADIAL****9.21.2 Function Documentation**

**9.21.2.1** `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

**9.21.2.2** `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

**9.21.2.3** `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

**9.21.2.4** `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

**9.21.2.5** void gslc\_ElemXRadialSetIndicator ( gslc\_tsGui \* *pGui*, gslc\_tsElemRef \* *pElemRef*, gslc\_tsColor *colGauge*, uint16\_t *nIndicLen*, uint16\_t *nIndicTip*, bool *blndicFill* )

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>blndicFill</i>	Fill in the indicator if true

**Returns**

none

**9.21.2.6** 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

**9.21.2.7** 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

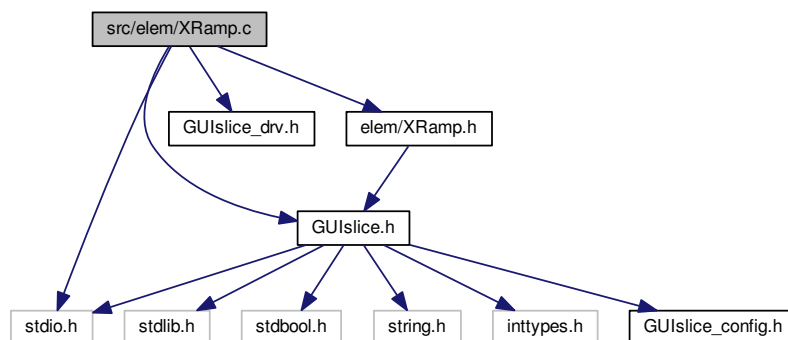
## Returns

none

## 9.22 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](#) []

## 9.22.1 Function Documentation

**9.22.1.1** `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 <i>nVal</i> comparison
in	<i>nMax</i>	Maximum value of gauge for <i>nVal</i> 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

**9.22.1.2** `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

**9.22.1.3** `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

**9.22.1.4** `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

**9.22.2 Variable Documentation**

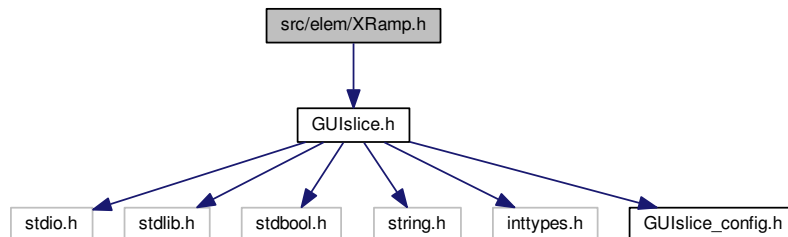
**9.22.2.1** `const char GSLC_PMEM_ERRSTR_NULL[]`

**9.22.2.2** `const char GSLC_PMEM_ERRSTR_PXD_NULL[]`

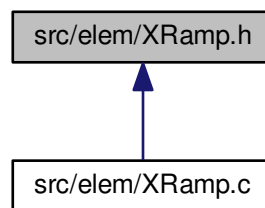
## 9.23 src/elem/XRamp.h File Reference

```
#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\_, col↵  
Frame\_, 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.*

### 9.23.1 Macro Definition Documentation

9.23.1.1 `#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

9.23.1.2 `#define GSLC_TYPEX_RAMP`

### 9.23.2 Function Documentation

9.23.2.1 `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

### 9.23.2.2 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

### 9.23.2.3 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

**9.23.2.4** 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**

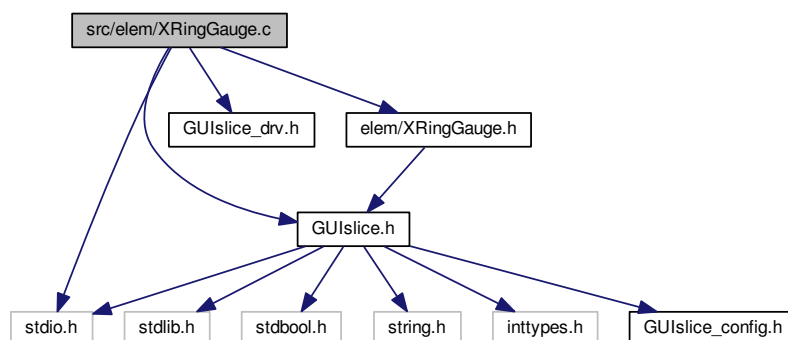
in	<i>pGui</i>	Pointer to GUI
in	<i>pElemRef</i>	Pointer to Element reference
in	<i>nVal</i>	New value to show in gauge

**Returns**

none

**9.24 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](#) []

### 9.24.1 Function Documentation

- 9.24.1.1 [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

**9.24.1.2** `bool gslc_ElemXRingGaugeDraw ( void * pvGui, void * pElemRef, gslc_teRedrawType eRedraw )`

Draw the template element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pElemRef</i>	Void ptr to Element (typecast to gslc_tsElemRef*)
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

**9.24.1.3** `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

**9.24.1.4** 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

**9.24.1.5** 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

**9.24.1.6** 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>collnactive</i>	Color of inactive region

**Returns**

none

**9.24.1.7** 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.

**Returns**

none

**9.24.1.8** 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

9.24.1.9 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

9.24.1.10 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

## 9.24.2 Variable Documentation

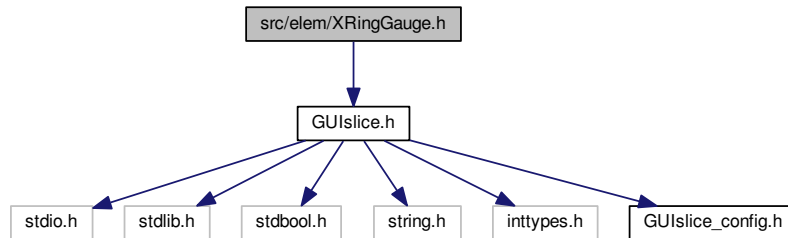
9.24.2.1 const char GSLC\_PMEM\_ERRSTR\_NULL[ ]

9.24.2.2 const char GSLC\_PMEM\_ERRSTR\_PXD\_NULL[ ]

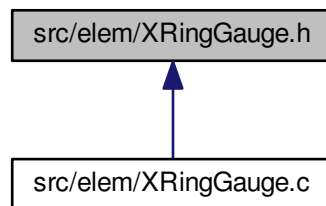
## 9.25 src/elem/XRingGauge.h File Reference

```
#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)
  - bool [gslc\\_ElemXRingGaugeDraw](#) (void \*pvGui, void \*pvElemRef, [gslc\\_teRedrawType](#) eRedraw)
- Create an XRingGauge element.*

*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.*

## 9.25.1 Macro Definition Documentation

### 9.25.1.1 #define GSLC\_TYPEX\_RING

### 9.25.1.2 #define XRING\_STR\_MAX

## 9.25.2 Function Documentation

### 9.25.2.1 [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

**9.25.2.2** `bool gslc_ElemXRingGaugeDraw ( void * pvGui, void * pvElemRef, gslc_teRedrawType eRedraw )`

Draw the template element on the screen.

- Called from [gslc\\_ElemDraw\(\)](#)

**Parameters**

in	<i>pGui</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

**9.25.2.3** `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

**9.25.2.4** 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

**9.25.2.5** 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

**9.25.2.6** 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>collnactive</i>	Color of inactive region

**Returns**

none

**9.25.2.7** 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.

**Returns**

none

**9.25.2.8** 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

9.25.2.9 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

9.25.2.10 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

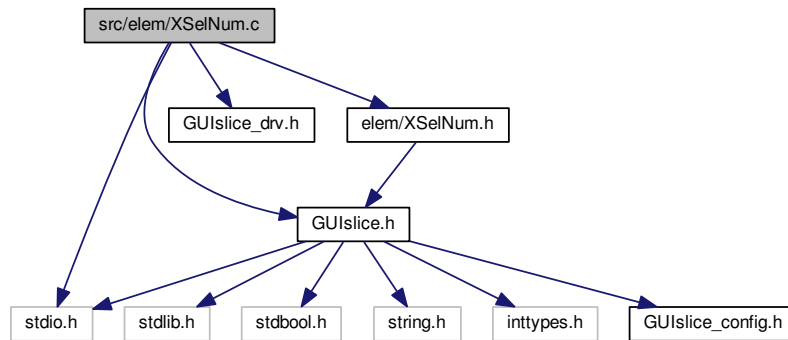
#### Returns

none

## 9.26 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:



## Variables

- const char [GSLC\\_PMEM ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM ERRSTR\\_PXD\\_NULL](#) []

### 9.26.1 Variable Documentation

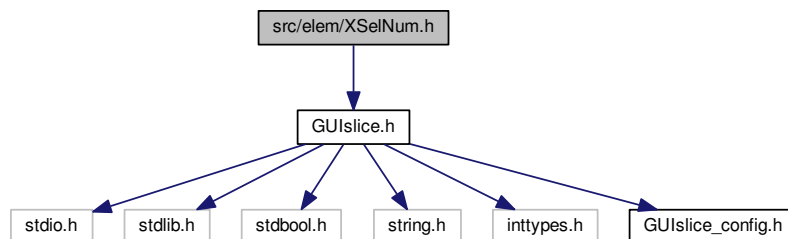
9.26.1.1 const char [GSLC\\_PMEM ERRSTR\\_NULL](#) []

9.26.1.2 const char [GSLC\\_PMEM ERRSTR\\_PXD\\_NULL](#) []

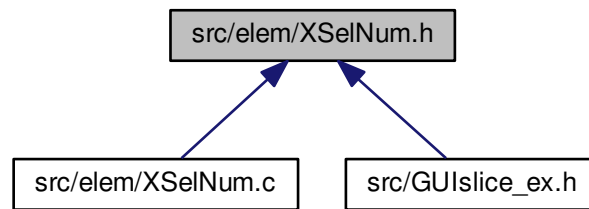
## 9.27 src/elem/XSelNum.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XSelNum.h:



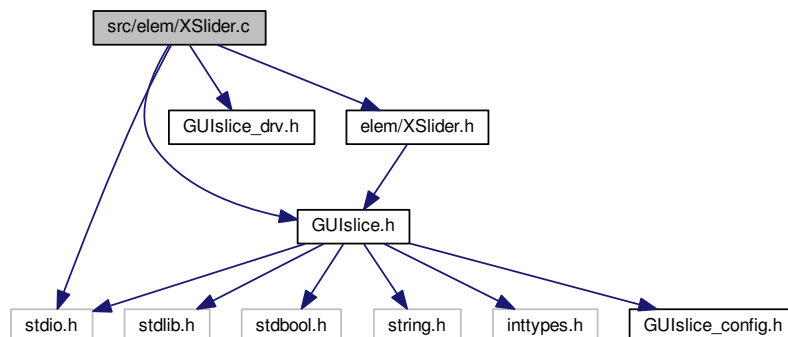
This graph shows which files directly or indirectly include this file:



## 9.28 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.
- `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](#) []

## 9.28.1 Function Documentation

9.28.1.1 [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 <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>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

9.28.1.2 [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>pElemRef</i>	Void ptr to Element (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

## Returns

true if success, false otherwise

**9.28.1.3** `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

**9.28.1.4** `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

**9.28.1.5** `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

**9.28.1.6** 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

**9.28.1.7** 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 ref (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

## 9.28.2 Variable Documentation

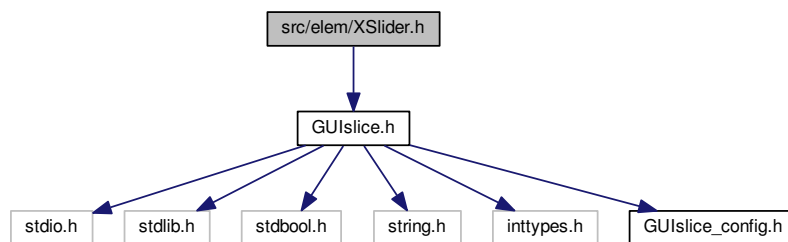
9.28.2.1 `const char GSLC_PMEM ERRSTR_NULL[]`

9.28.2.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

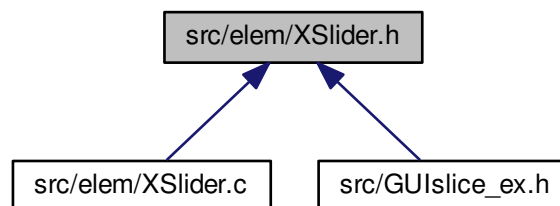
## 9.29 src/elem/XSlider.h File Reference

```
#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_tsX↔Slider *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 col↔Trim, uint16_t nTickDiv, int16_t nTickLen, gslc_tsColor colTick)`  
*Set a Slider element's current position.*
- `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.*

### 9.29.1 Macro Definition Documentation

- 9.29.1.1 `#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

## Parameters

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

## 9.29.1.2 #define GSLC\_TYPEX\_SLIDER

## 9.29.2 Typedef Documentation

## 9.29.2.1 typedef bool(\* GSLC\_CB\_XSLIDER\_POS)(void \*pvGui, void \*pvElem, int16\_t nPos)

Callback function for slider feedback.

## 9.29.3 Function Documentation

## 9.29.3.1 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

### 9.29.3.2 `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 (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

### 9.29.3.3 `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

### 9.29.3.4 `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

**9.29.3.5** 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

**9.29.3.6** 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

**9.29.3.7** 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

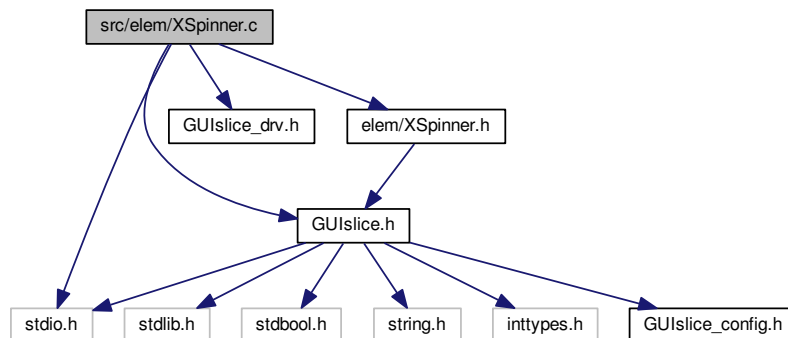
in	<i>pvGui</i>	Void ptr to GUI (typecast to gslc_tsGui*)
in	<i>pvElemRef</i>	Void ptr to Element ref (typecast to gslc_tsElemRef*)
in	<i>eTouch</i>	Touch event type
in	<i>nRelX</i>	Touch X coord relative to element
in	<i>nRelY</i>	Touch Y coord relative to element

## Returns

true if success, false otherwise

### 9.30 src/elem/XSpinner.c File Reference

```
#include "GUIslice.h"
#include "GUIslice_drv.h"
#include "elem/XSpinner.h"
#include <stdio.h>
Include dependency graph for XSpinner.c:
```



## Variables

- const char [GSLC\\_PMEM ERRSTR\\_NULL](#) []
- const char [GSLC\\_PMEM ERRSTR\\_PXD\\_NULL](#) []

#### 9.30.1 Variable Documentation

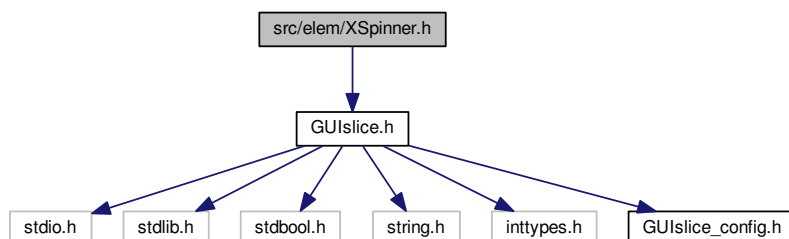
9.30.1.1 const char [GSLC\\_PMEM ERRSTR\\_NULL](#) []

9.30.1.2 const char [GSLC\\_PMEM ERRSTR\\_PXD\\_NULL](#) []

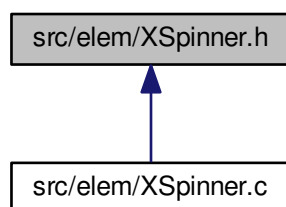
## 9.31 src/elem/XSpinner.h File Reference

```
#include "GUIslice.h"
```

Include dependency graph for XSpinner.h:



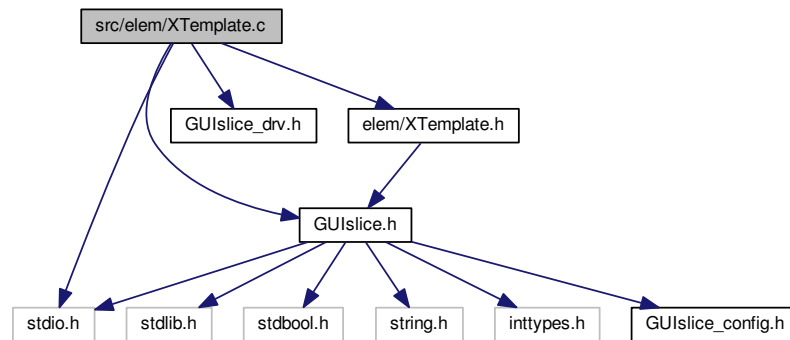
This graph shows which files directly or indirectly include this file:



## 9.32 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](#) []

### 9.32.1 Function Documentation

9.32.1.1 [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 <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 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

### 9.32.1.2 `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 (typecast to <code>gslc_tsElemRef*</code> )
in	<i>eRedraw</i>	Redraw mode

**Returns**

true if success, false otherwise

### 9.32.1.3 `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 ref (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

## 9.32.2 Variable Documentation

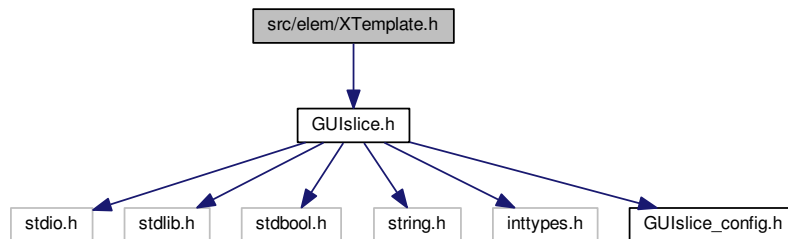
### 9.32.2.1 `const char GSLC_PMEM_ERRSTR_NULL[]`

### 9.32.2.2 `const char GSLC_PMEM_ERRSTR_PXD_NULL[]`

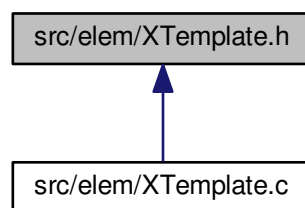
### 9.33 src/elem/XTemplate.h File Reference

```
#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\\_TYPEPEX\\_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.*

### 9.33.1 Macro Definition Documentation

#### 9.33.1.1 #define GSLC\_TYPEX\_TEMPLATE

### 9.33.2 Function Documentation

#### 9.33.2.1 `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

#### 9.33.2.2 `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 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

#### 9.33.2.3 `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 ref (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

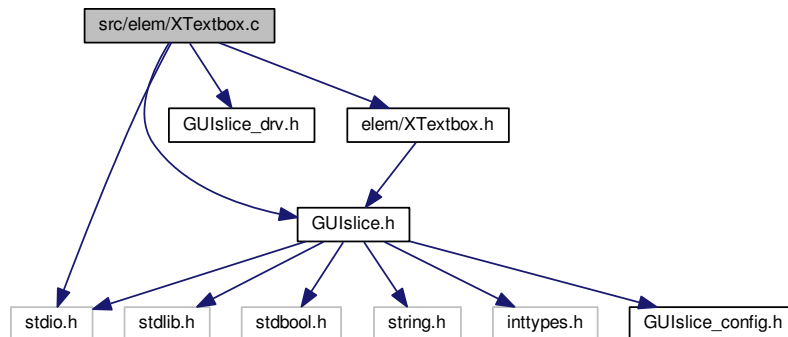
#### Returns

true if success, false otherwise

## 9.34 src/elem/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](#) []

## 9.34.1 Function Documentation

### 9.34.1.1 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

### 9.34.1.2 void [gslc\\_ElemXTextboxBufAdd](#) ( [gslc\\_tsGui](#) \* *pGui*, [gslc\\_tsElemRef](#) \* *pElemRef*, unsigned char *chNew*, bool *bAdvance* )

### 9.34.1.3 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

9.34.1.4 `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

9.34.1.5 `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 <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 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 ( <code>nBufRows*nBufCols</code> ) 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

9.34.1.6 `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 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

9.34.1.7 `void gslc_ElemXTextboxLineWrAdv ( gslc_tsGui * pGui, gslc_tsElemRef * pElemRef )`

9.34.1.8 `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

9.34.1.9 `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

9.34.1.10 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

**9.34.2 Variable Documentation**

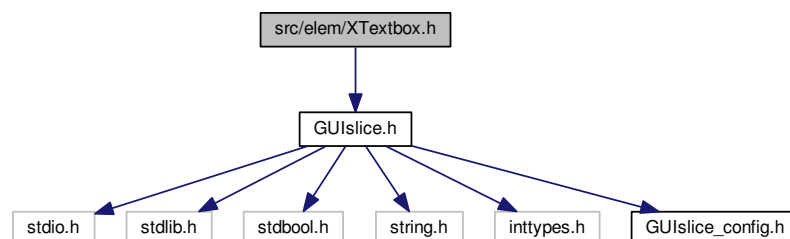
9.34.2.1 const char GSLC\_PMEM ERRSTR\_NULL[]

9.34.2.2 const char GSLC\_PMEM ERRSTR\_PXD\_NULL[]

**9.35 src/elem/XTextbox.h File Reference**

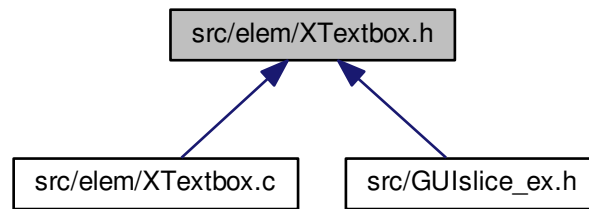
```
#include "GUIslice.h"
```

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.*

### 9.35.1 Macro Definition Documentation

9.35.1.1 `#define GSLC_TYPEX_TEXTBOX`

9.35.1.2 `#define GSLC_XTEXTBOX_CODE_COL_RESET`

9.35.1.3 `#define GSLC_XTEXTBOX_CODE_COL_SET`

Definitions for textbox special inline codes.

9.35.1.4 `#define XTEXTBOX_REDRAW_ALL`

9.35.1.5 `#define XTEXTBOX_REDRAW_NONE`

### 9.35.2 Function Documentation

9.35.2.1 `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

9.35.2.2 `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

9.35.2.3 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

9.35.2.4 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 <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 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 ( <i>nBufRows</i> * <i>nBufCols</i> ) 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

#### 9.35.2.5 `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

#### 9.35.2.6 `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

#### 9.35.2.7 `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

9.35.2.8 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

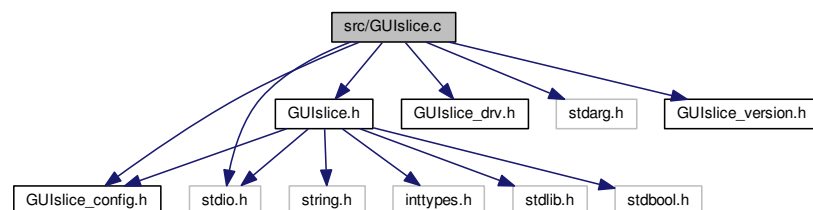
## Returns

none

## 9.36 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_teDebugPrintState` {  
`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)
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) eAction, int16\_t nActionVal)
- bool [gslc\\_InputMapLookup](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) \*peAction, int16\_t \*pnActionVal)
- 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\\_teEventType](#) 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)
- 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 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\\_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 nPageId)

*Find a page in the GUI by its ID.*
- [gslc\\_tsElemRef](#) \* [gslc\\_PageFindElemByld](#) ([gslc\\_tsGui](#) \*pGui, int16\_t nPageId, int16\_t nElemId)

*Find an element in the GUI by its Page ID and Element ID.*
- int16\_t [gslc\\_PageFocusStep](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage, bool bNext)
- int [gslc\\_ElemGetId](#) ([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.*
- [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](#) slmgRef, [gslc\\_tsImgRef](#) 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 r←Elem, 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_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 n←MarginY)`  
*Set the margin around of a textual element (X & Y offsets can be different)*
- `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_telemRedrawType` `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_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.*
- 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_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.*
- void `gslc_TrackInput` (`gslc_tsGui` \*pGui, `gslc_tsPage` \*pPage, `gslc_telInputRawEvent` 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_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 values.*

- void [gslc\\_SetTouchRemapYX](#) ([gslc\\_tsGui](#) \*pGui, bool bSwap)  
*Configure touchscreen XY swap.*
- [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.*
- bool [gslc\\_SetClipRect](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsRect](#) \*pRect)  
*Set the clipping rectangle for further drawing.*
- void [gslc\\_ElemSetImage](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElemRef](#) \*pElemRef, [gslc\\_tsImgRef](#) sImgRef, [gslc\\_tsImgRefSel](#) 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.*
- bool [gslc\\_CollectFindFocusStep](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsCollect](#) \*pCollect, bool bNext, bool \*pbWrapped, int16\_t \*pnElemInd)
- [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)`  
*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)`  
*Find an element in a collection by a coordinate coordinate.*
- `int16_t gslc_CollectGetFocus (gslc_tsGui *pGui, gslc_tsCollect *pCollect)`  
*Get the element index within a collection that is currently in focus.*
- `void gslc_CollectSetFocus (gslc_tsGui *pGui, gslc_tsCollect *pCollect, int16_t nElemInd)`  
*Set the element index within a collection that is currently in focus.*

## Variables

- `GSLC_CB_DEBUG_OUT g_pfDebugOut`  
*Global debug output function.*
- `uint16_t m_nLUTSinFOX16 [257]`
- `const char GSLC_PMEM_ERRSTR_NULL []`
- `const char GSLC_PMEM_ERRSTR_PXD_NULL []`

## 9.36.1 Enumeration Type Documentation

### 9.36.1.1 enum gslc\_teDebugPrintState

#### Enumerator

**`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`**

## 9.36.2 Function Documentation

- 9.36.2.1 `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 )`
- 9.36.2.2 `bool gslc_FontSetBase ( gslc_tsGui * pGui, uint8_t nFontInd, int16_t nFontId, gslc_teFontRefType eFontRefType, const void * pvFontRef, uint16_t nFontSz )`
- 9.36.2.3 `void gslc_OrderCoord ( int16_t * pnX0, int16_t * pnY0, int16_t * pnX1, int16_t * pnY1 )`
- 9.36.2.4 `void gslc_SwapCoords ( int16_t * pnXa, int16_t * pnYa, int16_t * pnXb, int16_t * pnYb )`

## 9.36.3 Variable Documentation

9.36.3.1 `const char ERRSTR_NULL[]`

9.36.3.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

9.36.3.3 `GSLC_CB_DEBUG_OUT g_pfDebugOut`

Global debug output function.

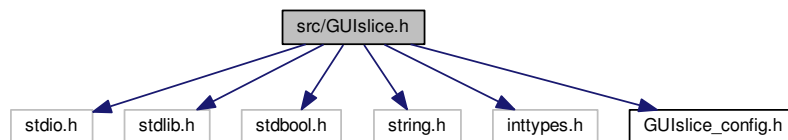
- The user assigns this function via [gslc\\_InitDebug\(\)](#)

9.36.3.4 `uint16_t m_nLUTSinF0X16`

## 9.37 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\\_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\\_2PI](#)
- #define [GSLC\\_ELEM\\_FEA\\_VALID](#)  
*Element features type.*
- #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_DK3`  
*Gray (dark)*
- #define `GSLC_COL_GRAY_DK2`  
*Gray (dark)*
- #define `GSLC_COL_GRAY_DK1`  
*Gray (dark)*
- #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_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 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_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\\_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.*
- 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\\_teAction](#) {  
  [GSLC\\_ACTION\\_UNDEF](#), [GSLC\\_ACTION\\_NONE](#), [GSLC\\_ACTION\\_FOCUS\\_PREV](#), [GSLC\\_ACTION\\_FOCUS\\_NEXT](#),  
  [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_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_EVTSUB_↵`  
`UB_DRAW_FORCE` }  
*Event sub-types.*
- enum `gslc_teRedrawType` { `GSLC_REDRAW_NONE`, `GSLC_REDRAW_FULL`, `GSLC_REDRAW_INC` }  
*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_GLOWING`,  
`GSLC_ELEMREF_VISIBLE`, `GSLC_ELEMREF_SRC`, `GSLC_ELEMREF_REDRAW_MASK` }  
*Element reference flags: Describes characteristics of an element.*
- enum `gslc_telmgRefFlags` {  
`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_IM_↵`  
`GREF_FMT_RAW1`,  
`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\\_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\\_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\\_tslmgRefFlags](#) eFmt)

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

  - [gslc\\_tslmgRef](#) [gslc\\_GetImageFromSD](#) (const char \*pFname, [gslc\\_tslmgRefFlags](#) eFmt)

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

  - [gslc\\_tslmgRef](#) [gslc\\_GetImageFromRam](#) (unsigned char \*plmgBuf, [gslc\\_tslmgRefFlags](#) eFmt)

*Create an image reference to a bitmap in SRAM.*

  - [gslc\\_tslmgRef](#) [gslc\\_GetImageFromProg](#) (const unsigned char \*plmgBuf, [gslc\\_tslmgRefFlags](#) 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 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.*

  - 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_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.*

- `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_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_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_tsTxtFlags 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.*
- 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.*
- 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 bOnlyClickEn)  
*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 values.*
- void [gslc\\_SetTouchRemapYX](#) ([gslc\\_tsGui](#) \*pGui, bool bSwap)  
*Configure touchscreen XY swap.*
- void [gslc\\_SetPinPollFunc](#) ([gslc\\_tsGui](#) \*pGui, [GSLC\\_CB\\_PIN\\_POLL](#) pfunc)
- void [gslc\\_InitInputMap](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsInputMap](#) \*asInputMap, uint8\_t nInputMapMax)
- void [gslc\\_InputMapAdd](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_telInputRawEvent](#) eInputEvent, int16\_t nInputVal, [gslc\\_teAction](#) eAction, int16\_t nActionVal)

- [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\\_teElemRefFlags](#) 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\\_teRedrawType](#) 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\\_teTxtFlags](#) 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.*
- [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.*
- [int16\\_t](#) [gslc\\_PageFocusStep](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsPage](#) \*pPage, [bool](#) bNext)
- [gslc\\_tsEvent](#) [gslc\\_EventCreate](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_teEventType](#) eType, [uint8\\_t](#) nSubType, [void](#) \*pvScope, [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` eTouch, 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` \*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)  
*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)  
*Set the element within a collection that is currently being tracked.*
  - int16\_t `gslc_CollectGetFocus` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect)  
*Get the element index within a collection that is currently in focus.*
  - void `gslc_CollectSetFocus` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, int16\_t nElemInd)  
*Set the element index within a collection that is currently in focus.*
  - bool `gslc_CollectFindFocusStep` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, bool bNext, bool \*pbWrapped, int16\_t \*pnElemInd)  
*Assign the parent element reference to all elements within a collection.*
  - void `gslc_CollectSetParent` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsElemRef` \*pElemRefParent)  
*Common event handler function for an element collection.*
  - bool `gslc_CollectEvent` (void \*pvGui, `gslc_tsEvent` sEvent)  
*Handle touch events within the element collection.*
  - void `gslc_CollectTouch` (`gslc_tsGui` \*pGui, `gslc_tsCollect` \*pCollect, `gslc_tsEventTouch` \*pEventTouch)  
*Handle dispatch of touch (up,down,move) events to compound elements sub elements.*
  - bool `gslc_CollectTouchCompound` (void \*pvGui, void \*pvElemRef, `gslc_teTouch` eTouch, int16\_t nRelX, int16\_t nRelY, `gslc_tsCollect` \*pCollect)  
*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_tsPage` \*pPage, `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)
  - 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.*

## 9.37.1 Macro Definition Documentation

### 9.37.1.1 `#define GSLC_2PI`

### 9.37.1.2 `#define GSLC_ALIGN_BOT_LEFT`

Align to bottom-left.

### 9.37.1.3 `#define GSLC_ALIGN_BOT_MID`

Align to middle of bottom.

### 9.37.1.4 `#define GSLC_ALIGN_BOT_RIGHT`

Align to bottom-right.

### 9.37.1.5 `#define GSLC_ALIGN_MID_LEFT`

Align to middle of left side.

### 9.37.1.6 `#define GSLC_ALIGN_MID_MID`

Align to center.

### 9.37.1.7 `#define GSLC_ALIGN_MID_RIGHT`

Align to middle of right side.

**9.37.1.8 #define GSLC\_ALIGN\_TOP\_LEFT**

Align to top-left.

**9.37.1.9 #define GSLC\_ALIGN\_TOP\_MID**

Align to middle of top.

**9.37.1.10 #define GSLC\_ALIGN\_TOP\_RIGHT**

Align to top-right.

**9.37.1.11 #define GSLC\_ALIGNH\_LEFT**

Horizontal align to left.

**9.37.1.12 #define GSLC\_ALIGNH\_MID**

Horizontal align to middle.

**9.37.1.13 #define GSLC\_ALIGNH\_RIGHT**

Horizontal align to right.

**9.37.1.14 #define GSLC\_ALIGNV\_BOT**

Vertical align to bottom.

**9.37.1.15 #define GSLC\_ALIGNV\_MID**

Vertical align to middle.

**9.37.1.16 #define GSLC\_ALIGNV\_TOP**

Element text alignment.

Vertical align to top

**9.37.1.17 #define GSLC\_COL\_BLACK**

Black.

9.37.1.18 `#define GSLC_COL_BLUE`

Blue.

9.37.1.19 `#define GSLC_COL_BLUE_DK1`

Blue (dark1)

9.37.1.20 `#define GSLC_COL_BLUE_DK2`

Blue (dark2)

9.37.1.21 `#define GSLC_COL_BLUE_DK3`

Blue (dark3)

9.37.1.22 `#define GSLC_COL_BLUE_DK4`

Blue (dark4)

9.37.1.23 `#define GSLC_COL_BLUE_LT1`

Blue (light1)

9.37.1.24 `#define GSLC_COL_BLUE_LT2`

Blue (light2)

9.37.1.25 `#define GSLC_COL_BLUE_LT3`

Blue (light3)

9.37.1.26 `#define GSLC_COL_BLUE_LT4`

Blue (light4)

9.37.1.27 `#define GSLC_COL_BROWN`

Brown.

9.37.1.28 `#define GSLC_COL_CYAN`

Cyan.

9.37.1.29 `#define GSLC_COL_GRAY`

Gray.

9.37.1.30 `#define GSLC_COL_GRAY_DK1`

Gray (dark)

9.37.1.31 `#define GSLC_COL_GRAY_DK2`

Gray (dark)

9.37.1.32 `#define GSLC_COL_GRAY_DK3`

Gray (dark)

9.37.1.33 `#define GSLC_COL_GRAY_LT1`

Gray (light1)

9.37.1.34 `#define GSLC_COL_GRAY_LT2`

Gray (light2)

9.37.1.35 `#define GSLC_COL_GRAY_LT3`

Gray (light3)

9.37.1.36 `#define GSLC_COL_GREEN`

Green.

9.37.1.37 `#define GSLC_COL_GREEN_DK1`

Green (dark1)

9.37.1.38 `#define GSLC_COL_GREEN_DK2`

Green (dark2)

9.37.1.39 `#define GSLC_COL_GREEN_DK3`

Green (dark3)

9.37.1.40 `#define GSLC_COL_GREEN_DK4`

Green (dark4)

9.37.1.41 `#define GSLC_COL_GREEN_LT1`

Green (light1)

9.37.1.42 `#define GSLC_COL_GREEN_LT2`

Green (light2)

9.37.1.43 `#define GSLC_COL_GREEN_LT3`

Green (light3)

9.37.1.44 `#define GSLC_COL_GREEN_LT4`

Green (light4)

9.37.1.45 `#define GSLC_COL_MAGENTA`

Magenta.

9.37.1.46 `#define GSLC_COL_ORANGE`

Orange.

9.37.1.47 `#define GSLC_COL_PURPLE`

Purple.



9.37.1.48 `#define GSLC_COL_RED`

Red.

9.37.1.49 `#define GSLC_COL_RED_DK1`

Red (dark1)

9.37.1.50 `#define GSLC_COL_RED_DK2`

Red (dark2)

9.37.1.51 `#define GSLC_COL_RED_DK3`

Red (dark3)

9.37.1.52 `#define GSLC_COL_RED_DK4`

Basic color definition.

Red (dark4)

9.37.1.53 `#define GSLC_COL_RED_LT1`

Red (light1)

9.37.1.54 `#define GSLC_COL_RED_LT2`

Red (light2)

9.37.1.55 `#define GSLC_COL_RED_LT3`

Red (light3)

9.37.1.56 `#define GSLC_COL_RED_LT4`

Red (light4)

9.37.1.57 `#define GSLC_COL_TEAL`

Teal.

9.37.1.58 `#define GSLC_COL_WHITE`

White.

9.37.1.59 `#define GSLC_COL_YELLOW`

Yellow.

9.37.1.60 `#define GSLC_COL_YELLOW_DK`

Yellow (dark)

9.37.1.61 `#define GSLC_COLMONO_BLACK`

Black.

9.37.1.62 `#define GSLC_COLMONO_WHITE`

White.

9.37.1.63 `#define GSLC_ELEM_FEA_CLICK_EN`

Element accepts touch presses.

9.37.1.64 `#define GSLC_ELEM_FEA_FILL_EN`

Element is drawn with a fill.

9.37.1.65 `#define GSLC_ELEM_FEA_FRAME_EN`

Element is drawn with a frame.

9.37.1.66 `#define GSLC_ELEM_FEA_GLOW_EN`

Element supports glowing state.

9.37.1.67 `#define GSLC_ELEM_FEA_NONE`

Element default (no features set))

9.37.1.68 `#define GSLC_ELEM_FEA_ROUND_EN`

Element is drawn with a rounded profile.

9.37.1.69 `#define GSLC_ELEM_FEA_VALID`

Element features type.

Element record is valid

9.37.1.70 `#define GSLC_ELEMREF_DEFAULT`

Define the default element reference flags for new elements.

9.37.1.71 `#define GSLC_PMEM`

## 9.37.2 Typedef Documentation

9.37.2.1 `typedef int16_t(* GSLC_CB_DEBUG_OUT)(char ch)`

9.37.2.2 `typedef bool(* GSLC_CB_DRAW)(void *pvGui, void *pvElemRef, gslc_teRedrawType eRedraw)`

Callback function for element drawing.

9.37.2.3 `typedef bool(* GSLC_CB_EVENT)(void *pvGui, gslc_tsEvent sEvent)`

Callback function for element drawing.

9.37.2.4 `typedef bool(* GSLC_CB_INPUT)(void *pvGui, void *pvElemRef, int16_t nStatus, void *pvData)`

Callback function for element input ready.

9.37.2.5 `typedef bool(* GSLC_CB_PIN_POLL)(void *pvGui, int16_t *pnPinInd, int16_t *pnPinVal)`

Callback function for pin polling.

9.37.2.6 `typedef bool(* GSLC_CB_TICK)(void *pvGui, void *pvElemRef)`

Callback function for element tick.

**9.37.2.7** `typedef bool(* GSLC_CB_TOUCH) (void *pvGui, void *pvElemRef, gslc_teTouch eTouch, int16_t nX, int16_t nY)`

Callback function for element touch tracking.

**9.37.2.8** `typedef struct gslc_tsColor gslc_tsColor`

Color structure. Defines RGB triplet.

**9.37.2.9** `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.

**9.37.2.10** `typedef struct gslc_tsEvent gslc_tsEvent`

Event structure.

**9.37.2.11** `typedef struct gslc_tsEventTouch gslc_tsEventTouch`

Structure used to pass touch data through event.

**9.37.2.12** `typedef struct gslc_tsPt gslc_tsPt`

Define point coordinates.

**9.37.2.13** `typedef struct gslc_tsRect gslc_tsRect`

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

### 9.37.3 Enumeration Type Documentation

#### 9.37.3.1 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\_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.

#### 9.37.3.2 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)

#### 9.37.3.3 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.

#### 9.37.3.4 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_GLOWING`** Element state is glowing.

**`GSLC_ELEMREF_VISIBLE`** Element is currently shown (ie. visible)

**`GSLC_ELEMREF_SRC`** Mask for Source flags.

**`GSLC_ELEMREF_REDRAW_MASK`** Mask for Redraw flags.

#### 9.37.3.5 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.

#### 9.37.3.6 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.

### 9.37.3.7 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.

### 9.37.3.8 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.

### 9.37.3.9 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.

9.37.3.10 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.

9.37.3.11 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\_SRC*** Mask for Source flags.

***GSLC\_IMGREF\_FMT*** Mask for Format flags.

9.37.3.12 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.

9.37.3.13 enum `gslc_telInputRawEvent`

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)



9.37.3.14 enum `gslc_tePageld`

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.

9.37.3.15 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)

9.37.3.16 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.

### 9.37.3.17 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.

### 9.37.3.18 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 inside 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\_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.

9.37.3.19 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`**

9.37.3.20 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.

## 9.37.4 Variable Documentation

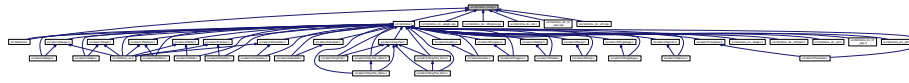
9.37.4.1 `GSLC_CB_DEBUG_OUT` `g_pfDebugOut`

Global debug output function.

- The user assigns this function via [`gslc\_InitDebug\(\)`](#)

### 9.38 src/GUISlice\_config.h File Reference

This graph shows which files directly or indirectly include this file:



### 9.39 src/GUISlice\_config\_ard.h File Reference

#### Macros

- `#define DRV_DISP_ADAGFX`
- `#define DRV_TOUCH_NONE`
- `#define DRV_DISP_ADAGFX_ILI9341`
- `#define ADAGFX_PIN_CS`
- `#define ADAGFX_PIN_DC`
- `#define ADAGFX_PIN_RST`
- `#define ADAGFX_PIN_SDCS`
- `#define ADAGFX_PIN_WR`
- `#define ADAGFX_PIN_RD`
- `#define ADAGFX_SPI_HW`
- `#define ADAGFX_PIN_MOSI`
- `#define ADAGFX_PIN_MISO`
- `#define ADAGFX_PIN_CLK`
- `#define GSLC_ROTATE`
- `#define TOUCH_ROTATION_DATA`
- `#define TOUCH_ROTATION_SWAPXY(rotation)`
- `#define TOUCH_ROTATION_FLIPX(rotation)`
- `#define TOUCH_ROTATION_FLIPY(rotation)`
- `#define ADATOUCH_SWAP_XY`
- `#define ADATOUCH_FLIP_X`
- `#define ADATOUCH_FLIP_Y`
- `#define GSLC_TOUCH_MAX_EVT`
- `#define DEBUG_ERR`
- `#define GSLC_FEATURE_COMPOUND`
- `#define GSLC_FEATURE_XGAUGE_RADIAL`
- `#define GSLC_FEATURE_XGAUGE_RAMP`
- `#define GSLC_FEATURE_XTEXTBOX_EMBED`
- `#define GSLC_FEATURE_INPUT`
- `#define GSLC_SD_EN`
- `#define GSLC_SD_BUFFPIXEL`
- `#define GSLC_CLIP_EN`
- `#define GSLC_BMP_TRANS_EN`
- `#define GSLC_BMP_TRANS_RGB`
- `#define GSLC_LOCAL_STR`
- `#define GSLC_LOCAL_STR_LEN`
- `#define GSLC_USE_FLOAT`
- `#define GSLC_DEV_TOUCH`
- `#define GSLC_USE_PROGMEM`

### 9.39.1 Macro Definition Documentation

9.39.1.1 `#define ADAGFX_PIN_CLK`

9.39.1.2 `#define ADAGFX_PIN_CS`

9.39.1.3 `#define ADAGFX_PIN_DC`

9.39.1.4 `#define ADAGFX_PIN_MISO`

9.39.1.5 `#define ADAGFX_PIN_MOSI`

9.39.1.6 `#define ADAGFX_PIN_RD`

9.39.1.7 `#define ADAGFX_PIN_RST`

9.39.1.8 `#define ADAGFX_PIN_SDCS`

9.39.1.9 `#define ADAGFX_PIN_WR`

9.39.1.10 `#define ADAGFX_SPI_HW`

9.39.1.11 `#define ADATOUCH_FLIP_X`

9.39.1.12 `#define ADATOUCH_FLIP_Y`

9.39.1.13 `#define ADATOUCH_SWAP_XY`

9.39.1.14 `#define DEBUG_ERR`

9.39.1.15 `#define DRV_DISP_ADAGFX`

9.39.1.16 `#define DRV_DISP_ADAGFX_ILI9341`

9.39.1.17 `#define DRV_TOUCH_NONE`

9.39.1.18 `#define GSLC_BMP_TRANS_EN`

9.39.1.19 `#define GSLC_BMP_TRANS_RGB`

9.39.1.20 `#define GSLC_CLIP_EN`

9.39.1.21 `#define GSLC_DEV_TOUCH`

9.39.1.22 `#define GSLC_FEATURE_COMPOUND`

```

9.39.1.23 #define GSLC_FEATURE_INPUT

9.39.1.24 #define GSLC_FEATURE_XGAUGE_RADIAL

9.39.1.25 #define GSLC_FEATURE_XGAUGE_RAMP

9.39.1.26 #define GSLC_FEATURE_XTEXTBOX_EMBED

9.39.1.27 #define GSLC_LOCAL_STR

9.39.1.28 #define GSLC_LOCAL_STR_LEN

9.39.1.29 #define GSLC_ROTATE

9.39.1.30 #define GSLC_SD_BUFFPIXEL

9.39.1.31 #define GSLC_SD_EN

9.39.1.32 #define GSLC_TOUCH_MAX_EVT

9.39.1.33 #define GSLC_USE_FLOAT

9.39.1.34 #define GSLC_USE_PROGMEM

9.39.1.35 #define TOUCH_ROTATION_DATA

9.39.1.36 #define TOUCH_ROTATION_FLIPX( rotation )

9.39.1.37 #define TOUCH_ROTATION_FLIPY( rotation )

9.39.1.38 #define TOUCH_ROTATION_SWAPXY( rotation )

```

## 9.40 src/GUISlice\_config\_linux.h File Reference

### Macros

- #define [DRV\\_DISP\\_SDL1](#)
- #define [DRV\\_TOUCH\\_TSLIB](#)
- #define [GSLC\\_FEATURE\\_COMPOUND](#)
- #define [GSLC\\_FEATURE\\_XGAUGE\\_RADIAL](#)
- #define [GSLC\\_FEATURE\\_XGAUGE\\_RAMP](#)
- #define [GSLC\\_FEATURE\\_XTEXTBOX\\_EMBED](#)
- #define [GSLC\\_FEATURE\\_INPUT](#)
- #define [DEBUG\\_ERR](#)
- #define [GSLC\\_DEV\\_FB](#)
- #define [GSLC\\_DEV\\_TOUCH](#)
- #define [GSLC\\_DEV\\_VID\\_DRV](#)
- #define [DRV\\_SDL\\_FIX\\_START](#)
- #define [DRV\\_SDL\\_MOUSE\\_SHOW](#)
- #define [GSLC\\_LOCAL\\_STR](#)
- #define [GSLC\\_USE\\_FLOAT](#)
- #define [ADATOUCH\\_SWAP\\_XY](#)
- #define [ADATOUCH\\_FLIP\\_X](#)
- #define [ADATOUCH\\_FLIP\\_Y](#)
- #define [GSLC\\_TOUCH\\_MAX\\_EVT](#)
- #define [GSLC\\_LOCAL\\_STR\\_LEN](#)
- #define [GSLC\\_BMP\\_TRANS\\_EN](#)
- #define [GSLC\\_BMP\\_TRANS\\_RGB](#)
- #define [GSLC\\_USE\\_PROGMEM](#)

### 9.40.1 Macro Definition Documentation

9.40.1.1 `#define ADATOUCH_FLIP_X`

9.40.1.2 `#define ADATOUCH_FLIP_Y`

9.40.1.3 `#define ADATOUCH_SWAP_XY`

9.40.1.4 `#define DEBUG_ERR`

9.40.1.5 `#define DRV_DISP_SDL1`

9.40.1.6 `#define DRV_SDL_FIX_START`

9.40.1.7 `#define DRV_SDL_MOUSE_SHOW`

9.40.1.8 `#define DRV_TOUCH_TSLIB`

9.40.1.9 `#define GSLC_BMP_TRANS_EN`

9.40.1.10 `#define GSLC_BMP_TRANS_RGB`

9.40.1.11 `#define GSLC_DEV_FB`

9.40.1.12 `#define GSLC_DEV_TOUCH`

9.40.1.13 `#define GSLC_DEV_VID_DRV`

9.40.1.14 `#define GSLC_FEATURE_COMPOUND`

9.40.1.15 `#define GSLC_FEATURE_INPUT`

9.40.1.16 `#define GSLC_FEATURE_XGAUGE_RADIAL`

9.40.1.17 `#define GSLC_FEATURE_XGAUGE_RAMP`

9.40.1.18 `#define GSLC_FEATURE_XTEXTBOX_EMBED`

9.40.1.19 `#define GSLC_LOCAL_STR`

9.40.1.20 `#define GSLC_LOCAL_STR_LEN`

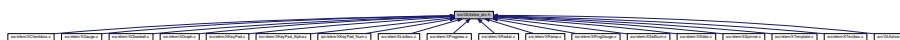
9.40.1.21 `#define GSLC_TOUCH_MAX_EVT`

9.40.1.22 `#define GSLC_USE_FLOAT`

9.40.1.23 `#define GSLC_USE_PROGMEM`

## 9.41 src/GUISlice\_drv.h File Reference

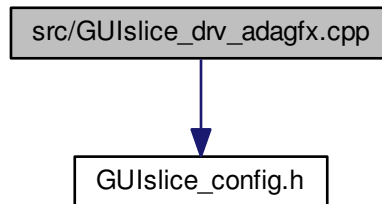
This graph shows which files directly or indirectly include this file:



## 9.42 src/GUISlice\_drv\_adagfx.cpp File Reference

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_adagfx.cpp:



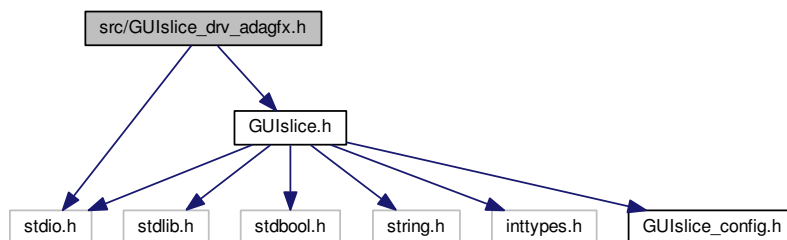
## 9.43 src/GUISlice\_drv\_adagfx.h File Reference

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_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_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)

### 9.43.1 Detailed Description

GUISlice library (driver layer for Adafruit-GFX)

### 9.43.2 Macro Definition Documentation

#### 9.43.2.1 #define DRV\_HAS\_DRAW\_CIRCLE\_FILL

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.43.2.2 #define DRV\_HAS\_DRAW\_CIRCLE\_FRAME

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.43.2.3 #define DRV\_HAS\_DRAW\_LINE

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.43.2.4 #define DRV\_HAS\_DRAW\_POINT

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.43.2.5 #define DRV\_HAS\_DRAW\_POINTS

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.43.2.6 #define DRV\_HAS\_DRAW\_RECT\_FILL

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.43.2.7 #define DRV\_HAS\_DRAW\_RECT\_FRAME

Support [gslc\\_DrvDrawFrameRect\(\)](#)

9.43.2.8 `#define DRV_HAS_DRAW_RECT_ROUND_FILL`

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

9.43.2.9 `#define DRV_HAS_DRAW_RECT_ROUND_FRAME`

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

9.43.2.10 `#define DRV_HAS_DRAW_TEXT`

Support [gslc\\_DrvDrawTxt\(\)](#)

9.43.2.11 `#define DRV_HAS_DRAW_TRI_FILL`

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

9.43.2.12 `#define DRV_HAS_DRAW_TRI_FRAME`

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

9.43.2.13 `#define DRV_OVERRIDE_TXT_ALIGN`

Driver provides text alignment.

### 9.43.3 Function Documentation

9.43.3.1 `uint16_t gslc_DrvAdaptColorToRaw ( gslc_tsColor nCol )`

9.43.3.2 `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

## 9.43.3.3 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

## 9.43.3.4 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

## 9.43.3.5 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

**9.43.3.6** `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

**9.43.3.7** `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

**9.43.3.8** `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

**9.43.3.9** `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

**9.43.3.10** `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

**9.43.3.11** `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

**9.43.3.12** `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

**9.43.3.13** `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

**9.43.3.14** `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
----	-------------	----------------



## Parameters

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

9.43.3.15 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

9.43.3.16 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

9.43.3.17 `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>nNumPt</i>	Number of points in array
in	<i>nCol</i>	Color RGB value to draw

#### Returns

true if success, false if error

9.43.3.18 `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

9.43.3.19 `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

**9.43.3.20 void gslc\_DrvFontsDestruct ( gslc\_tsGui \* *pGui* )**

Release all fonts defined in the GUI.

**Parameters**

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

**Returns**

none

**9.43.3.21 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.

**9.43.3.22 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.

**9.43.3.23** `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

**9.43.3.24** `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

**9.43.3.25** `bool gslc_DrvGetTouch ( gslc_tsGui * pGui, int16_t * pnX, int16_t * pnY, uint16_t * pnPress, gslc_telnputRawEvent * pnInputEvent, 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>pnInputEvent</i>	Indication of event type
out	<i>pnInputVal</i>	Additional data for event type

**Returns**

true if an event was detected or false otherwise

9.43.3.26 `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

9.43.3.27 `void gslc_DrvImageDestruct ( void * pvlmg )`

Release an image surface.

#### Parameters

in	<i>pvlmg</i>	Void ptr to image
----	--------------	-------------------

#### Returns

none

9.43.3.28 `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

**9.43.3.29**    `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

**9.43.3.30**    `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

**9.43.3.31**    `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

**9.43.3.32 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

**9.43.3.33 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

**9.43.3.34 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

#### 9.43.3.35 `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

#### 9.43.3.36 `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

#### 9.43.3.37 `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

**9.43.3.38** `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

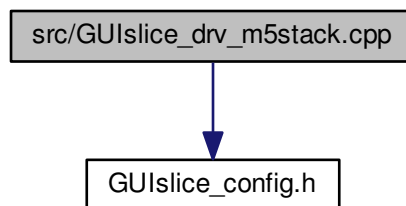
**Returns**

true if success, false if error

## 9.44 src/GUISlice\_drv\_m5stack.cpp File Reference

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_m5stack.cpp:

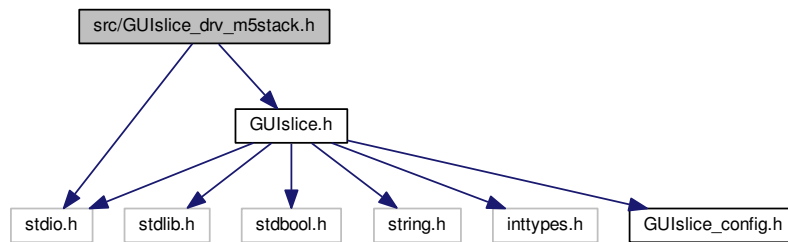


## 9.45 src/GUISlice\_drv\_m5stack.h File Reference

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_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](#) 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](#) 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](#) []

## 9.45.1 Detailed Description

GUIslice library (driver layer for M5stack)

## 9.45.2 Macro Definition Documentation

### 9.45.2.1 #define DRV\_HAS\_DRAW\_CIRCLE\_FILL

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.45.2.2 #define DRV\_HAS\_DRAW\_CIRCLE\_FRAME

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.45.2.3 #define DRV\_HAS\_DRAW\_LINE

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.45.2.4 #define DRV\_HAS\_DRAW\_POINT

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.45.2.5 #define DRV\_HAS\_DRAW\_POINTS

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.45.2.6 #define DRV\_HAS\_DRAW\_RECT\_FILL

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.45.2.7 #define DRV\_HAS\_DRAW\_RECT\_FRAME

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.45.2.8 #define DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.45.2.9 #define DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.45.2.10 #define DRV\_HAS\_DRAW\_TEXT

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.45.2.11 #define DRV\_HAS\_DRAW\_TRI\_FILL

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

9.45.2.12 `#define DRV_HAS_DRAW_TRI_FRAME`

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

9.45.2.13 `#define DRV_OVERRIDE_TXT_ALIGN`

Driver provides text alignment.

### 9.45.3 Function Documentation

9.45.3.1 `uint16_t gslc_DrvAdaptColorToRaw ( gslc_tsColor nCol )`

9.45.3.2 `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

9.45.3.3 `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

9.45.3.4 `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

**9.45.3.5** `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

**9.45.3.6** `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

**9.45.3.7** `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

**9.45.3.8** `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

**9.45.3.9** `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
----	-------------	----------------



## Parameters

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

9.45.3.10 `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

9.45.3.11 `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

9.45.3.12 `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

**9.45.3.13** `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.

**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>slmgRef</i>	Image reference

**Returns**

true if success, false if fail

**9.45.3.14** `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

**9.45.3.15** 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

**9.45.3.16** 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

**9.45.3.17** 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
----	-------------	----------------

## Parameters

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

9.45.3.18 `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

9.45.3.19 `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

**9.45.3.20** `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

**9.45.3.21** `void gslc_DrvFontsDestruct ( gslc_tsGui * pGui )`

Release all fonts defined in the GUI.

**Parameters**

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

**Returns**

none

**9.45.3.22** `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.

#### 9.45.3.23 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.

#### 9.45.3.24 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

#### 9.45.3.25 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

#### 9.45.3.26 bool gslc\_DrvGetTxtSize ( gslc\_tsGui \* pGui, gslc\_tsFont \* pFont, const char \* pStr, gslc\_teTextFlags 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

9.45.3.27 void gslc\_DrvImageDestruct ( void \* *pvImg* )

Release an image surface.

## Parameters

in	<i>pvImg</i>	Void ptr to image
----	--------------	-------------------

## Returns

none

9.45.3.28 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

#### 9.45.3.29 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

#### 9.45.3.30 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

#### 9.45.3.31 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



**9.45.3.32** `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

**9.45.3.33** `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

**9.45.3.34** `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

#### 9.45.3.35 `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

#### 9.45.3.36 `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

#### 9.45.3.37 `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

### 9.45.4 Variable Documentation

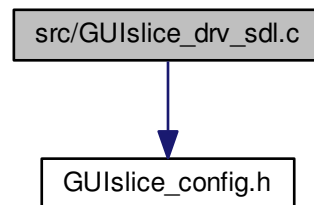
#### 9.45.4.1 `const char GSLC_PMEM ERRSTR_NULL[]`

9.45.4.2 `const char GSLC_PMEM ERRSTR_PXD_NULL[]`

## 9.46 src/GUISlice\_drv\_sdl.c File Reference

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_sdl.c:



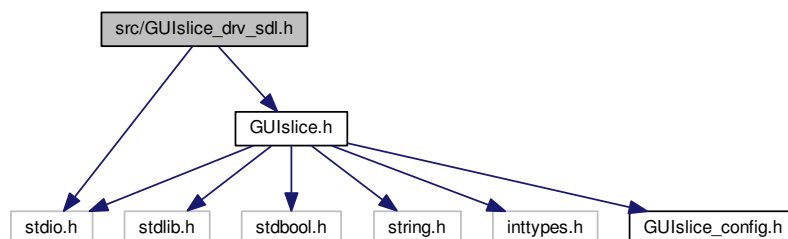
## 9.47 src/GUISlice\_drv\_sdl.h File Reference

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](#) 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.*
- 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\\_telInputRawEvent](#) \*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.*

### 9.47.1 Detailed Description

GUISlice library (driver layer for LINUX / SDL)

### 9.47.2 Macro Definition Documentation

#### 9.47.2.1 #define DRV\_HAS\_DRAW\_POINT

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.47.2.2 #define DRV\_OVERRIDE\_TXT\_ALIGN

Driver provides text alignment.

### 9.47.3 Function Documentation

#### 9.47.3.1 SDL\_Color [gslc\\_DrvAdaptColor](#) ( [gslc\\_tsColor](#) sCol )

Translate a [gslc\\_tsColor](#) into an SDL\_Color.

Parameters

in	sCol	<a href="#">gslc_tsColor</a>
----	------	------------------------------

**Returns**

Converted SDL\_Color

**9.47.3.2 SDL\_Rect gslc\_DrvAdaptRect ( gslc\_tsRect *rRect* )**

Translate a [gslc\\_tsRect](#) into an SDL\_Rect.

**Parameters**

in	<i>rRect</i>	<a href="#">gslc_tsRect</a>
----	--------------	-----------------------------

**Returns**

Converted SDL\_Rect

**9.47.3.3 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

**9.47.3.4 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

9.47.3.5 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

9.47.3.6 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

9.47.3.7 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

9.47.3.8 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

**9.47.3.9** `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

**9.47.3.10** `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



9.47.3.11 `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

9.47.3.12 `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

9.47.3.13 `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>pvFontRef</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

#### 9.47.3.14 void gslc\_DrvFontsDestruct ( gslc\_tsGui \* pGui )

Release all fonts defined in the GUI.

##### Parameters

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

##### Returns

none

#### 9.47.3.15 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.

#### 9.47.3.16 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.

9.47.3.17 `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

**9.47.3.18** `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

**9.47.3.19** `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

**9.47.3.20** `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
----	-------------	----------------

**Parameters**

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

**9.47.3.21 void gslc\_DrvImageDestruct ( void \* *pvlmg* )**

Release an image surface.

**Parameters**

in	<i>pvlmg</i>	Void ptr to image
----	--------------	-------------------

**Returns**

none

**9.47.3.22 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

**9.47.3.23 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

**9.47.3.24 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

**9.47.3.25 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

#### 9.47.3.26 void gslc\_DrvReportInfoPost ( )

Report driver debug info (after initialization)

##### Returns

none

#### 9.47.3.27 void gslc\_DrvReportInfoPre ( )

Report driver debug info (before initialization)

##### Returns

none

#### 9.47.3.28 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

#### 9.47.3.29 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



### 9.47.3.30 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

### 9.47.3.31 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

### 9.47.3.32 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

### 9.47.3.33 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

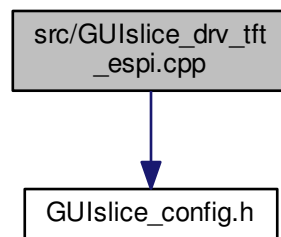
**Returns**

true if success, false if error

**9.48 src/GUISlice\_drv\_tft\_espi.cpp File Reference**

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_tft\_espi.cpp:

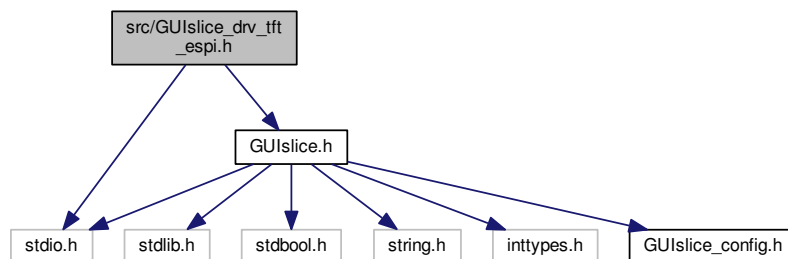
**9.49 src/GUISlice\_drv\_tft\_espi.h File Reference**

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 [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\\_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](#) 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\\_tsImgRef](#) slmgRef)

*Set an element's normal-state image.*
- bool [gslc\\_DrvSetElemImageGlow](#) ([gslc\\_tsGui](#) \*pGui, [gslc\\_tsElem](#) \*pElem, [gslc\\_tsImgRef](#) 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](#) 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)

### 9.49.1 Detailed Description

GUISlice library (driver layer for TFT-eSPI)

### 9.49.2 Macro Definition Documentation

#### 9.49.2.1 #define DRV\_HAS\_DRAW\_CIRCLE\_FILL

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.49.2.2 #define DRV\_HAS\_DRAW\_CIRCLE\_FRAME

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.49.2.3 #define DRV\_HAS\_DRAW\_LINE

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.49.2.4 #define DRV\_HAS\_DRAW\_POINT

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.49.2.5 #define DRV\_HAS\_DRAW\_POINTS

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.49.2.6 `#define DRV_HAS_DRAW_RECT_FILL`

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.49.2.7 `#define DRV_HAS_DRAW_RECT_FRAME`

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.49.2.8 `#define DRV_HAS_DRAW_RECT_ROUND_FILL`

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.49.2.9 `#define DRV_HAS_DRAW_RECT_ROUND_FRAME`

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.49.2.10 `#define DRV_HAS_DRAW_TEXT`

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.49.2.11 `#define DRV_HAS_DRAW_TRI_FILL`

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

#### 9.49.2.12 `#define DRV_HAS_DRAW_TRI_FRAME`

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

#### 9.49.2.13 `#define DRV_OVERRIDE_TXT_ALIGN`

Driver provides text alignment.

### 9.49.3 Function Documentation

#### 9.49.3.1 `uint16_t gslc_DrvAdaptColorToRaw ( gslc_tsColor nCol )`

#### 9.49.3.2 `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 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 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 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

**9.49.3.6** `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

**9.49.3.7** `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

**9.49.3.8** `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

**9.49.3.9** `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

**9.49.3.10** `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

9.49.3.11 `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

9.49.3.12 `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

9.49.3.13 `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

9.49.3.14 `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

9.49.3.15 `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

9.49.3.16 `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

9.49.3.17 `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

9.49.3.18 `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

9.49.3.19 `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

9.49.3.20 `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

9.49.3.21 `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 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 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 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** `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** `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

**9.49.3.27** `void gslc_DrvImageDestruct ( void * pVImg )`

Release an image surface.

**Parameters**

in	<i>pVImg</i>	Void ptr to image
----	--------------	-------------------

**Returns**

none

**9.49.3.28 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

**9.49.3.29 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



9.49.3.30 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

9.49.3.31 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

9.49.3.32 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

9.49.3.33 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

9.49.3.34 `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

9.49.3.35 `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

9.49.3.36 `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

9.49.3.37 `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

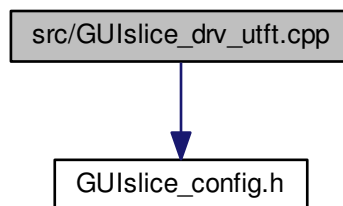
**Returns**

true if success, false if error

## 9.50 src/GUISlice\_drv\_utft.cpp File Reference

```
#include "GUISlice_config.h"
```

Include dependency graph for GUISlice\_drv\_utft.cpp:

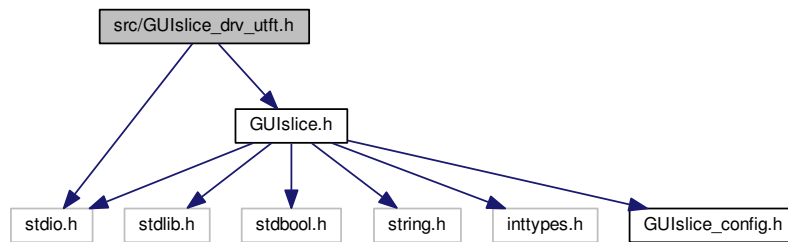


## 9.51 src/GUISlice\_drv\_utft.h File Reference

GUISlice library (driver layer for UTFT)

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

Include dependency graph for GUIslice\_drv\_utf8.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_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\\_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](#) \*pInputEvent, 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)

### 9.51.1 Detailed Description

GUIslice library (driver layer for UTFT)

### 9.51.2 Macro Definition Documentation

#### 9.51.2.1 #define DRV\_HAS\_DRAW\_CIRCLE\_FILL

Support [gslc\\_DrvDrawFillCircle\(\)](#)

#### 9.51.2.2 #define DRV\_HAS\_DRAW\_CIRCLE\_FRAME

Support [gslc\\_DrvDrawFrameCircle\(\)](#)

#### 9.51.2.3 #define DRV\_HAS\_DRAW\_LINE

Support [gslc\\_DrvDrawLine\(\)](#)

#### 9.51.2.4 #define DRV\_HAS\_DRAW\_POINT

Support [gslc\\_DrvDrawPoint\(\)](#)

#### 9.51.2.5 #define DRV\_HAS\_DRAW\_POINTS

Support [gslc\\_DrvDrawPoints\(\)](#)

#### 9.51.2.6 #define DRV\_HAS\_DRAW\_RECT\_FILL

Support [gslc\\_DrvDrawFillRect\(\)](#)

#### 9.51.2.7 #define DRV\_HAS\_DRAW\_RECT\_FRAME

Support [gslc\\_DrvDrawFrameRect\(\)](#)

#### 9.51.2.8 #define DRV\_HAS\_DRAW\_RECT\_ROUND\_FILL

Support [gslc\\_DrvDrawFillRoundRect\(\)](#)

#### 9.51.2.9 #define DRV\_HAS\_DRAW\_RECT\_ROUND\_FRAME

Support [gslc\\_DrvDrawFrameRoundRect\(\)](#)

#### 9.51.2.10 #define DRV\_HAS\_DRAW\_TEXT

Support [gslc\\_DrvDrawTxt\(\)](#)

#### 9.51.2.11 #define DRV\_HAS\_DRAW\_TRI\_FILL

Support [gslc\\_DrvDrawFillTriangle\(\)](#)

9.51.2.12 `#define DRV_HAS_DRAW_TRI_FRAME`

Support [gslc\\_DrvDrawFrameTriangle\(\)](#)

9.51.2.13 `#define DRV_OVERRIDE_TXT_ALIGN`

Driver provides text alignment.

### 9.51.3 Function Documentation

9.51.3.1 `uint16_t gslc_DrvAdaptColorToRaw ( gslc_tsColor nCol )`

9.51.3.2 `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 `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 `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** `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** `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** `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** `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** `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
----	-------------	----------------

## Parameters

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 `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 `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 `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** `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.

**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>slmgRef</i>	Image reference

**Returns**

true if success, false if fail

**9.51.3.14** `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** 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** 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** 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
----	-------------	----------------

## Parameters

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

9.51.3.18 `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

9.51.3.19 `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

#### 9.51.3.20 void gslc\_DrvFontsDestruct ( gslc\_tsGui \* pGui )

Release all fonts defined in the GUI.

##### Parameters

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

##### Returns

none

#### 9.51.3.21 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.

#### 9.51.3.22 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.

9.51.3.23 `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

**9.51.3.24** `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

**9.51.3.25** `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

**9.51.3.26** `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
----	-------------	----------------

**Parameters**

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 void gslc\_DrvImageDestruct ( void \* *pvlmg* )**

Release an image surface.

**Parameters**

in	<i>pvlmg</i>	Void ptr to image
----	--------------	-------------------

**Returns**

none

**9.51.3.28 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** `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

**9.51.3.30** `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

**9.51.3.31** `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

### 9.51.3.32 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

### 9.51.3.33 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

### 9.51.3.34 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

### 9.51.3.35 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

**9.51.3.36** `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

**9.51.3.37** `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

**9.51.3.38** `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

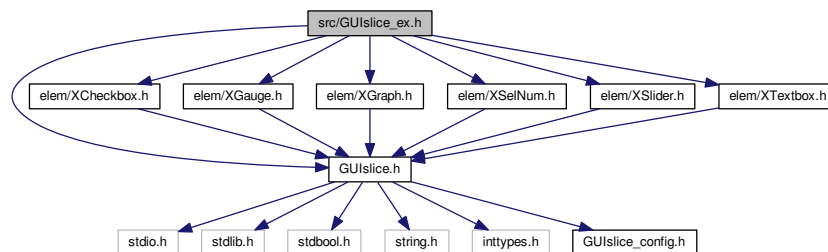
## Returns

true if success, false if error

## 9.52 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"
```

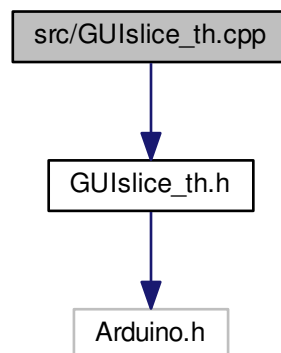
Include dependency graph for GUISlice\_ex.h:



## 9.53 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`

### 9.53.1 Function Documentation

9.53.1.1 `TouchHandler*` `gslc_getTouchHandler` (`void` )

9.53.1.2 `void` `gslc_InitTouchHandler` (`TouchHandler *` *pTH* )

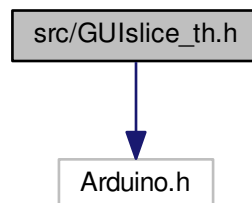
### 9.53.2 Variable Documentation

9.53.2.1 `TouchHandler*` `pTouchHandler`

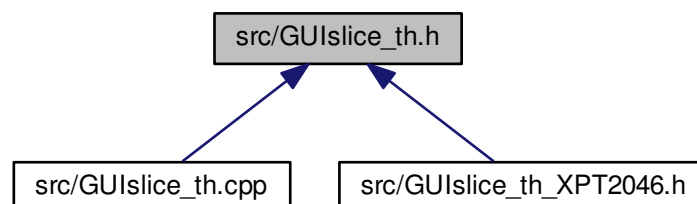
## 9.54 src/GUIslice\_th.h File Reference

```
#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)

### 9.54.1 Function Documentation

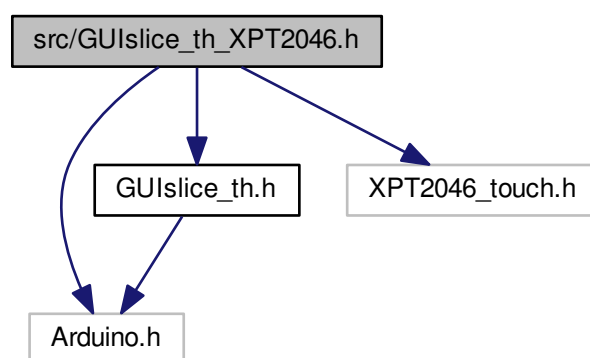
9.54.1.1 [TouchHandler](#)\* [gslc\\_getTouchHandler](#) ( void )

9.54.1.2 void [gslc\\_InitTouchHandler](#) ( [TouchHandler](#) \* *pTHO* )

## 9.55 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:



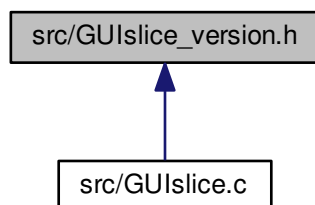
## Data Structures

- class [TouchHandler\\_XPT2046](#)



## 9.56 src/GUISlice\_version.h File Reference

This graph shows which files directly or indirectly include this file:



### Macros

- `#define` [GUISLICE\\_VER](#)

#### 9.56.1 Macro Definition Documentation

##### 9.56.1.1 `#define` GUISLICE\_VER

