TouchGFX v4.12.3 User's Manual

(C)opyright STMicroelectronics 2014-2019

www.st.com www.touchgfx.com

Contents

1	Cha	ngelog			1
2	Glob	oals			21
3	Nam	espace	Index		23
	3.1	Names	pace List		23
4	Hier	archical	Index		25
	4.1	Class I	Hierarchy		25
5	Clas	s Index			31
	5.1	Class I	_ist		31
6	Nam	espace	Documer	utation	41
	6.1	touchg	fx Namesp	ace Reference	41
		6.1.1	Typedef [Documentation	54
			6.1.1.1	Alignment	54
			6.1.1.2	EasingEquation	54
			6.1.1.3	Rendering Variant	55
			6.1.1.4	TextDirection	55
		6.1.2	Enumera	tion Type Documentation	55
			6.1.2.1	BlitOperations	55
			6.1.2.2	Direction	55
			6.1.2.3	DisplayOrientation	56
			6.1.2.4	DisplayRotation	56
			6.1.2.5	DMAType	56
			6.1.2.6	FrameBuffer	56
			6.1.2.7	GlyphFlags	57
			6.1.2.8	Gradient	57
			6.1.2.9	TextRotation	57
			6.1.2.10	WideTextAction	58
		6.1.3	Function	Documentation	58
			6191	abe/\	E0

iv CONTENTS

			6.1.3.2	ceil28_4()	58
			6.1.3.3	clz()	59
			6.1.3.4	finalizeTransition()	59
			6.1.3.5	fixed28_4Mul()	59
			6.1.3.6	fixed28_4ToFloat()	60
			6.1.3.7	floatToFixed16_16()	60
			6.1.3.8	floatToFixed28_4()	60
			6.1.3.9	floorDivMod()	60
			6.1.3.10	FrameBufferAllocatorSignalBlockDrawn()	61
			6.1.3.11	FrameBufferAllocatorWaitOnTransfer()	61
			6.1.3.12	gcd()	61
			6.1.3.13	hw_init()	61
			6.1.3.14	LCD2getPixel() [1/2]	62
			6.1.3.15	LCD2getPixel() [2/2]	62
			6.1.3.16	LCD2setPixel() [1/2]	62
			6.1.3.17	LCD2setPixel() [2/2]	62
			6.1.3.18	LCD2shiftVal()	63
			6.1.3.19	LCD4getPixel() [1/2]	63
			6.1.3.20	LCD4getPixel() [2/2]	63
			6.1.3.21	LCD4setPixel() [1/2]	64
			6.1.3.22	LCD4setPixel() [2/2]	64
			6.1.3.23	lookupBilinearRenderVariant()	64
			6.1.3.24	lookupNearestNeighborRenderVariant()	64
			6.1.3.25	makeTransition()	65
			6.1.3.26	memset()	65
			6.1.3.27	muldiv()	66
			6.1.3.28	operator*() [1/2]	66
			6.1.3.29	operator*() [2/2]	67
			6.1.3.30	prepareTransition()	67
			6.1.3.31	touchgfx_generic_init()	67
			6.1.3.32	touchgfx_init()	68
		6.1.4	Variable	Documentation	68
			6.1.4.1	TYPED_TEXT_INVALID	68
7	Class	s Docui	mentation		69
	7.1			lass Reference	69
		7.1.1	Detailed	Description	69
		7.1.2	Construc	tor & Destructor Documentation	70
			7.1.2.1	AbstractButton()	70
		7.1.3	Member	Function Documentation	70

CONTENTS

		7.1.3.1	getPressedState()	70
		7.1.3.2	getType()	70
		7.1.3.3	handleClickEvent()	70
		7.1.3.4	setAction()	71
7.2	Abstra	ctButtonCo	ontainer Class Reference	71
	7.2.1	Detailed I	Description	72
	7.2.2	Member I	Function Documentation	72
		7.2.2.1	getAlpha()	72
		7.2.2.2	getPressed()	72
		7.2.2.3	setAction()	72
		7.2.2.4	setAlpha()	73
		7.2.2.5	setPressed()	73
7.3	Abstra	ctClock Cla	ass Reference	73
	7.3.1	Construc	tor & Destructor Documentation	74
		7.3.1.1	AbstractClock()	74
		7.3.1.2	~AbstractClock()	74
	7.3.2	Member I	Function Documentation	74
		7.3.2.1	getCurrentHour()	74
		7.3.2.2	getCurrentMinute()	74
		7.3.2.3	getCurrentSecond()	75
		7.3.2.4	setTime12Hour()	75
		7.3.2.5	setTime24Hour()	75
		7.3.2.6	updateClock()	75
7.4	Abstra	ctDirection	Progress Class Reference	76
	7.4.1	Detailed I	Description	76
	7.4.2	Member I	Enumeration Documentation	76
		7.4.2.1	DirectionType	76
	7.4.3	Construc	tor & Destructor Documentation	76
		7.4.3.1	AbstractDirectionProgress()	77
		7.4.3.2	~AbstractDirectionProgress()	77
	7.4.4	Member I	Function Documentation	77
		7.4.4.1	getDirection()	77
		7.4.4.2	setDirection()	77
7.5	Abstra	ctPainter C	Class Reference	77
	7.5.1	Detailed I	Description	78
	7.5.2	Construc	tor & Destructor Documentation	78
		7.5.2.1	AbstractPainter()	78
		7.5.2.2	~AbstractPainter()	78
	7.5.3	Member I	Function Documentation	78
		7.5.3.1	compatibleFramebuffer()	78

vi CONTENTS

		7.5.3.2	render()	79
		7.5.3.3	setOffset()	79
		7.5.3.4	setWidgetAlpha()	80
7.6	Abstrac	ctPainterA	BGR2222 Class Reference	80
	7.6.1	Detailed	Description	81
	7.6.2	Member	Function Documentation	81
		7.6.2.1	mixColors() [1/2]	81
		7.6.2.2	mixColors() [2/2]	81
		7.6.2.3	render()	82
		7.6.2.4	renderInit()	82
		7.6.2.5	renderNext()	82
		7.6.2.6	renderPixel()	83
7.7	Abstrac	ctPainterA	RGB2222 Class Reference	83
	7.7.1	Detailed	Description	84
	7.7.2	Member	Function Documentation	84
		7.7.2.1	mixColors() [1/2]	84
		7.7.2.2	mixColors() [2/2]	84
		7.7.2.3	render()	85
		7.7.2.4	renderInit()	85
		7.7.2.5	renderNext()	86
		7.7.2.6	renderPixel()	86
7.8	Abstrac	ctPainterA	RGB8888 Class Reference	86
	7.8.1	Detailed	Description	87
	7.8.2	Member	Function Documentation	87
		7.8.2.1	render()	87
		7.8.2.2	renderInit()	88
		7.8.2.3	renderNext()	88
		7.8.2.4	renderPixel() [1/2]	88
		7.8.2.5	renderPixel() [2/2]	89
7.9	Abstrac	ctPainterB	GRA2222 Class Reference	89
	7.9.1	Detailed	Description	90
	7.9.2	Member	Function Documentation	90
		7.9.2.1	mixColors() [1/2]	90
		7.9.2.2	mixColors() [2/2]	90
		7.9.2.3	render()	91
		7.9.2.4	renderInit()	91
		7.9.2.5	renderNext()	92
		7.9.2.6	renderPixel()	92
7.10	Abstrac	ctPainterB'	W Class Reference	92
	7.10.1	Detailed	Description	93

CONTENTS vii

	7.10.2	Member Function Documentation
		7.10.2.1 render()
		7.10.2.2 renderInit()
		7.10.2.3 renderNext()
7.11	Abstrac	stPainterGRAY2 Class Reference
	7.11.1	Detailed Description
	7.11.2	Member Function Documentation
		7.11.2.1 render()
		7.11.2.2 renderInit()
		7.11.2.3 renderNext()
		7.11.2.4 renderPixel()
7.12	Abstrac	ctPainterGRAY4 Class Reference
	7.12.1	Detailed Description
	7.12.2	Member Function Documentation
		7.12.2.1 render()
		7.12.2.2 renderInit()
		7.12.2.3 renderNext()
		7.12.2.4 renderPixel()
7.13	Abstrac	ctPainterRGB565 Class Reference
	7.13.1	Detailed Description
	7.13.2	Member Function Documentation
		7.13.2.1 mixColors() [1/2]
		7.13.2.2 mixColors() [2/2]
		7.13.2.3 render()
		7.13.2.4 renderInit()
		7.13.2.5 renderNext()
		7.13.2.6 renderPixel()
7.14	Abstrac	ctPainterRGB888 Class Reference
	7.14.1	Detailed Description
	7.14.2	Member Function Documentation
		7.14.2.1 render()
		7.14.2.2 renderInit()
		7.14.2.3 renderNext()
		7.14.2.4 renderPixel()
7.15	Abstrac	ctPainterRGBA2222 Class Reference
	7.15.1	Detailed Description
	7.15.2	Member Function Documentation
		7.15.2.1 mixColors() [1/2]
		7.15.2.2 mixColors() [2/2]
		7.15.2.3 render()

viii CONTENTS

		7.15.2.4 renderInit()
		7.15.2.5 renderNext()
		7.15.2.6 renderPixel()
7.16	Abstrac	ctPartition Class Reference
	7.16.1	Detailed Description
	7.16.2	Constructor & Destructor Documentation
		7.16.2.1 ~AbstractPartition()
		7.16.2.2 AbstractPartition()
	7.16.3	Member Function Documentation
		7.16.3.1 allocate() [1/2]
		7.16.3.2 allocate() [2/2]
		7.16.3.3 allocateAt() [1/2]
		7.16.3.4 allocateAt() [2/2]
		7.16.3.5 at() [1/2]
		7.16.3.6 at() [2/2]
		7.16.3.7 capacity()
		7.16.3.8 clear()
		7.16.3.9 dec()
		7.16.3.10 element() [1/2]
		7.16.3.11 element() [2/2]
		7.16.3.12 element_size()
		7.16.3.13 find()
		7.16.3.14 getAllocationCount()
		7.16.3.15 indexOf()
7.17	Abstrac	ctProgressIndicator Class Reference
	7.17.1	Detailed Description
	7.17.2	Constructor & Destructor Documentation
		7.17.2.1 AbstractProgressIndicator()
		7.17.2.2 ~AbstractProgressIndicator()
	7.17.3	Member Function Documentation
		7.17.3.1 getProgress()
		7.17.3.2 getProgressIndicatorHeight()
		7.17.3.3 getProgressIndicatorWidth()
		7.17.3.4 getProgressIndicatorX()
		7.17.3.5 getProgressIndicatorY()
		7.17.3.6 getRange() [1/3]
		7.17.3.7 getRange() [2/3]
		7.17.3.8 getRange() [3/3]
		7.17.3.9 getValue()
		7.17.3.10 setBackground()

CONTENTS

	7.17.3.11 setProgressIndicatorPosition()	18
	7.17.3.12 setRange()	19
	7.17.3.13 setValue()	19
7.18 Abstra	actShape Class Reference	20
7.18.1	Detailed Description	21
7.18.2	2 Constructor & Destructor Documentation	21
	7.18.2.1 AbstractShape()	22
	7.18.2.2 ~AbstractShape()	22
7.18.3	Member Function Documentation	22
	7.18.3.1 drawCanvasWidget()	22
	7.18.3.2 getAngle() [1/2]	22
	7.18.3.3 getAngle() [2/2]	23
	7.18.3.4 getCacheX()	23
	7.18.3.5 getCacheY()	23
	7.18.3.6 getCornerX()	23
	7.18.3.7 getCornerY()	24
	7.18.3.8 getMinimalRect()	24
	7.18.3.9 getNumPoints()	24
	7.18.3.10 getOrigin()	25
	7.18.3.11 getScale()	25
	7.18.3.12 moveOrigin()	25
	7.18.3.13 setAngle()	26
	7.18.3.14 setCache()	26
	7.18.3.15 setCorner()	27
	7.18.3.16 setOrigin()	27
	7.18.3.17 setScale() [1/2]	28
	7.18.3.18 setScale() [2/2]	28
	7.18.3.19 setShape() [1/2]	29
	7.18.3.20 setShape() [2/2]	29
	7.18.3.21 updateAbstractShapeCache()	29
	7.18.3.22 updateAngle()	30
	7.18.3.23 updateScale()	30
7.19 Analo	gClock Class Reference	30
7.19.1	Detailed Description	32
7.19.2	2 Member Function Documentation	32
	7.19.2.1 animationEnabled()	32
	7.19.2.2 convertHandValueToAngle()	33
	7.19.2.3 getAnimationDuration()	33
	7.19.2.4 getHourHandMinuteCorrection()	33
	7.19.2.5 getMinuteHandSecondCorrection()	33

CONTENTS

		7.19.2.6 ir	nitializeTime12Hour()	34
		7.19.2.7 ir	nitializeTime24Hour()	34
		7.19.2.8 s	etAnimation()	34
		7.19.2.9 s	etBackground() [1/2]	35
		7.19.2.10 s	etBackground() [2/2]	35
		7.19.2.11 s	etHourHandMinuteCorrection()	35
		7.19.2.12 s	etMinuteHandSecondCorrection()	36
		7.19.2.13 s	etRotationCenter()	36
		7.19.2.14 s	etupHand()	36
		7.19.2.15 s	etupHourHand()	37
		7.19.2.16 s	etupMinuteHand()	37
		7.19.2.17 s	etupSecondHand()	37
		7.19.2.18 u	pdateClock()	38
7.20	Animat	edImage Cla	ass Reference	38
			escription	
	7.20.2		* & Destructor Documentation	
			nimatedImage() [1/2]	
			.nimatedImage() [2/2]	
	7.20.3		nction Documentation	
		_	etType()	
			andleTickEvent()	
			sAnimatedImageRunning()	
			Reverse()	
			Running()	
		•	auseAnimation()	
			etBitmap()	
			etBitmaps()	
			etDoneAction()	
			etUpdateTicksInterval()	
			tartAnimation()	
			topAnimation()	
7.21			tonStyle < T > Class Template Reference	
			escription	
	7.21.2		nction Documentation	
			etBitmaps()	
			etBitmapXY()	
7.55			etUpdateTicksInterval()	
			apper::AnimationSetting Struct Reference	
7.23			apper Class Reference	
	7.23.1	Detailed De	scription	46

CONTENTS xi

	7.23.2	Member E	Enumeration Documentation	146
		7.23.2.1	AnimationParameter	146
		7.23.2.2	AnimationState	147
	7.23.3	Construct	or & Destructor Documentation	147
		7.23.3.1	AnimationTextureMapper()	147
		7.23.3.2	\sim AnimationTextureMapper()	147
	7.23.4	Member F	Function Documentation	147
		7.23.4.1	cancelAnimationTextureMapperAnimation()	147
		7.23.4.2	getAnimationStep()	147
		7.23.4.3	handleTickEvent()	147
		7.23.4.4	isTextureMapperAnimationRunning()	148
		7.23.4.5	setTextureMapperAnimationEndedAction()	148
		7.23.4.6	$set Texture Mapper Animation Step Action () \\ \ \ldots \\ $	148
		7.23.4.7	setupAnimation()	148
		7.23.4.8	startAnimation()	149
7.24	Applica	tion Class	Reference	149
	7.24.1	Detailed D	Description	151
	7.24.2	Construct	or & Destructor Documentation	151
		7.24.2.1	Application()	151
	7.24.3	Member F	Function Documentation	151
		7.24.3.1	appSwitchScreen()	151
		7.24.3.2	cacheDrawOperations()	152
		7.24.3.3	clearAllTimerWidgets()	152
		7.24.3.4	draw() [1/2]	152
		7.24.3.5	draw() [2/2]	152
		7.24.3.6	getCurrentScreen()	152
		7.24.3.7	getDebugPrinter()	153
		7.24.3.8	getInstance()	153
		7.24.3.9	getNumberOfRegisteredTimerWidgets()	153
		7.24.3.10	getTimerWidgetCountForDrawable()	153
		7.24.3.11	handleClickEvent()	154
		7.24.3.12	handleDragEvent()	154
		7.24.3.13	handleGestureEvent()	154
		7.24.3.14	handleKeyEvent()	154
		7.24.3.15	handlePendingScreenTransition()	155
		7.24.3.16	handleTickEvent()	155
			invalidateArea()	
		7.24.3.18	registerTimerWidget()	155
			setDebugPrinter()	
		7.24.3.20	setDebugString()	156

xii CONTENTS

		7.24.3.21	switchScreen()		 	 	 	 	156
		7.24.3.22	unregisterTimerWidg	jet()	 	 	 	 	156
	7.24.4	Member [ata Documentation .		 	 	 	 	157
		7.24.4.1	instance		 	 	 	 	157
		7.24.4.2	MAX_TIMER_WIDG	ETS	 	 	 	 	157
7.25	Bitmap	Class Refe	rence		 	 	 	 	157
	7.25.1	Detailed [escription		 	 	 	 	159
	7.25.2	Member E	numeration Docume	ntation .	 	 	 	 	159
		7.25.2.1	BitmapFormat		 	 	 	 	159
		7.25.2.2	ClutFormat		 	 	 	 	160
	7.25.3	Construct	or & Destructor Docu	mentation	 	 	 	 	160
		7.25.3.1	Bitmap()		 	 	 	 	160
	7.25.4	Member F	unction Documentati	on	 	 	 	 	160
		7.25.4.1	cache()		 	 	 	 	160
		7.25.4.2	cacheAll()		 	 	 	 	161
		7.25.4.3	cacheGetAddress()		 	 	 	 	161
		7.25.4.4	cachelsCached()		 	 	 	 	161
		7.25.4.5	cacheRemoveBitma	o()	 	 	 	 	162
		7.25.4.6	cacheReplaceBitmap	o()	 	 	 	 	162
		7.25.4.7	clearCache()		 	 	 	 	162
		7.25.4.8	compactCache()		 	 	 	 	163
		7.25.4.9	dynamicBitmapAddS	iolidRect()	 	 	 	 	163
		7.25.4.10	dynamicBitmapCrea	ie()	 	 	 	 	163
		7.25.4.11	dynamicBitmapDelet	e()	 	 	 	 	164
		7.25.4.12	dynamicBitmapGetA	ddress()	 	 	 	 	164
		7.25.4.13	dynamicBitmapSetS	olidRect()	 	 	 	 	164
		7.25.4.14	getAlphaData()		 	 	 	 	165
		7.25.4.15	getCacheTopAddres	s()	 	 	 	 	165
		7.25.4.16	getData()		 	 	 	 	165
		7.25.4.17	getExtraData()		 	 	 	 	166
		7.25.4.18	getFormat()		 	 	 	 	166
		7.25.4.19	getHeight()		 	 	 	 	166
		7.25.4.20	getId()		 	 	 	 	166
		7.25.4.21	getRect()		 	 	 	 	167
		7.25.4.22	getSolidRect()		 	 	 	 	167
		7.25.4.23	getWidth()		 	 	 	 	167
		7.25.4.24	hasTransparentPixel	s()	 	 	 	 	167
		7.25.4.25	isAlphaPerPixel()		 	 	 	 	167
		7.25.4.26	operator"!=()		 	 	 	 	168
		7.25.4.27	operator==()		 	 	 	 	168

CONTENTS xiii

		7.25.4.28 registerBitmapDatabase()
		7.25.4.29 removeCache()
		7.25.4.30 setCache()
7.26	Bitmap	::BitmapData Struct Reference
	7.26.1	Detailed Description
	7.26.2	Member Function Documentation
		7.26.2.1 getFormat()
7.27	BlitOp	Struct Reference
	7.27.1	Detailed Description
	7.27.2	Member Data Documentation
		7.27.2.1 dstFormat
		7.27.2.2 operation
		7.27.2.3 srcFormat
7.28	Box Cla	ass Reference
	7.28.1	Detailed Description
	7.28.2	Constructor & Destructor Documentation
		7.28.2.1 Box() [1/2]
		7.28.2.2 Box() [2/2]
		7.28.2.3 ~Box()
	7.28.3	Member Function Documentation
		7.28.3.1 draw()
		7.28.3.2 forceReportAsSolid()
		7.28.3.3 getAlpha()
		7.28.3.4 getColor()
		7.28.3.5 getSolidRect()
		7.28.3.6 getType()
		7.28.3.7 setAlpha()
		7.28.3.8 setColor()
	7.28.4	Member Data Documentation
		7.28.4.1 reportAsSolid
7.29	BoxPro	gress Class Reference
	7.29.1	Detailed Description
	7.29.2	Constructor & Destructor Documentation
		7.29.2.1 BoxProgress()
		7.29.2.2 ~BoxProgress()
	7.29.3	Member Function Documentation
		7.29.3.1 getAlpha()
		7.29.3.2 getColor()
		7.29.3.3 setAlpha()
		7.29.3.4 setColor()

XIV

		7.29.3.5	setProgressIndicatorPosition()	178
		7.29.3.6	setValue()	178
7.30	BoxWit	hBorder C	class Reference	178
	7.30.1	Construc	tor & Destructor Documentation	179
		7.30.1.1	BoxWithBorder()	179
	7.30.2	Member	Function Documentation	180
		7.30.2.1	draw()	180
		7.30.2.2	getAlpha()	180
		7.30.2.3	getBorderColor()	180
		7.30.2.4	getBorderSize()	180
		7.30.2.5	getColor()	180
		7.30.2.6	getSolidRect()	181
		7.30.2.7	getType()	181
		7.30.2.8	setAlpha()	181
		7.30.2.9	setBorderColor()	181
		7.30.2.10) setBorderSize()	181
		7.30.2.11	setColor()	182
7.31	BoxWit	hBorderBu	uttonStyle < T > Class Template Reference	182
	7.31.1	Detailed	Description	183
	7.31.2	Member	Function Documentation	183
		7.31.2.1	setBorderSize()	183
		7.31.2.2	setBoxWithBorderColors()	183
		7.31.2.3	setBoxWithBorderHeight()	184
		7.31.2.4	setBoxWithBorderPosition()	184
		7.31.2.5	setBoxWithBorderWidth()	184
7.32	Button	Class Ref	erence	184
	7.32.1	Detailed	Description	185
	7.32.2	Construc	tor & Destructor Documentation	185
			Button()	
		7.32.2.2	~Button()	185
	7.32.3	Member	Function Documentation	186
		7.32.3.1	draw()	186
		7.32.3.2	getAlpha()	186
		7.32.3.3	getCurrentlyDisplayedBitmap()	186
		7.32.3.4	getSolidRect()	186
		7.32.3.5	getType()	
		7.32.3.6	setAlpha()	
			setBitmaps()	
7.33			Class Reference	
	7.33.1	Detailed	Description	188

CONTENTS xv

	7.33.2	Constructor & Destructor Documentation	 188
		7.33.2.1 \sim ButtonController()	 188
	7.33.3	Member Function Documentation	 188
		7.33.3.1 init()	 188
		7.33.3.2 reset()	 188
		7.33.3.3 sample()	 188
7.34	Buttons	Class Reference	 189
	7.34.1	Member Function Documentation	 189
		7.34.1.1 init()	 189
		7.34.1.2 sample()	 189
7.35	Button\	VithIcon Class Reference	 189
	7.35.1	Detailed Description	 190
	7.35.2	Member Function Documentation	 190
		7.35.2.1 draw()	 190
		$7.35.2.2 \hspace{0.2cm} getCurrentlyDisplayedIcon() \hspace{0.2cm} . 0.2cm$	 191
		7.35.2.3 $getIconX()$	 191
		7.35.2.4 getlconY()	 191
		7.35.2.5 getType()	 191
		7.35.2.6 setBitmaps()	 191
		7.35.2.7 setlconX()	 192
		7.35.2.8 setlconXY()	 192
		7.35.2.9 $setIconY()$	 192
7.36	Button\	VithLabel Class Reference	 193
	7.36.1	Detailed Description	 194
	7.36.2	Constructor & Destructor Documentation	 194
		7.36.2.1 ButtonWithLabel()	 194
	7.36.3	Member Function Documentation	 194
		7.36.3.1 draw()	 194
		$7.36.3.2 \hspace{0.2cm} \textbf{getLabelRotation()} \hspace{0.2cm} \dots \hspace{0.2cm} \dots \hspace{0.2cm} \dots$	 194
		7.36.3.3 getLabelText()	 194
		7.36.3.4 getSolidRect()	 195
		7.36.3.5 getType()	 195
		7.36.3.6 setLabelColor()	 195
		7.36.3.7 setLabelColorPressed() 	 195
		7.36.3.8 setLabelRotation()	 196
		7.36.3.9 setLabelText()	 196
		7.36.3.10 updateTextPosition()	 196
7.37	Cachea	bleContainer Class Reference	 196
	7.37.1	Detailed Description	 197
	7.37.2	Constructor & Destructor Documentation	 197

xvi CONTENTS

		7.37.2.1 CacheableContainer()	3 7
		7.37.2.2 ~CacheableContainer()	97
	7.37.3	Member Function Documentation	98
		7.37.3.1 enableCachedMode()	98
		7.37.3.2 invalidateRect()	98
		7.37.3.3 isChildInvalidated()	98
		7.37.3.4 setCacheBitmap()	98
		7.37.3.5 setupDrawChain()) 9
		7.37.3.6 updateCache() [1/2]) 9
		7.37.3.7 updateCache() [2/2]) 9
7.38	Cachea	bleContainer::CachedImage Class Reference	99
	7.38.1	Detailed Description)0
	7.38.2	Constructor & Destructor Documentation)0
		7.38.2.1 CachedImage())0
		7.38.2.2 ~CachedImage())0
	7.38.3	Member Function Documentation)0
		7.38.3.1 setupDrawChain())0
7.39	Bitmap	:CacheTableEntry Struct Reference)0
	7.39.1	Detailed Description)1
7.40	Callbac	<< dest_type, T1, T2, T3 > Struct Template Reference)1
	7.40.1	Detailed Description)1
	7.40.2	Constructor & Destructor Documentation)2
		7.40.2.1 Callback() [1/2])2
		7.40.2.2 Callback() [2/2])2
	7.40.3	Member Function Documentation)2
		7.40.3.1 execute())2
		7.40.3.2 isValid())2
7.41	Callbac	<< dest_type, T1, T2, void > Struct Template Reference)3
	7.41.1	Detailed Description)3
	7.41.2	Constructor & Destructor Documentation)3
		7.41.2.1 Callback() [1/2])4
		7.41.2.2 Callback() [2/2]	
	7.41.3	Member Function Documentation)4
		7.41.3.1 execute()	
		7.41.3.2 isValid())4
7.42	Callbac	<< dest_type, T1, void, void > Struct Template Reference)4
		Detailed Description	
	7.42.2	Constructor & Destructor Documentation	
		7.42.2.1 Callback() [1/2]	
		7.42.2.2 Callback() [2/2])5

CONTENTS xvii

	7.42.3	Member Function Documentation	206
		7.42.3.1 execute()	206
		7.42.3.2 isValid()	206
7.43	Callbac	k< dest_type, void, void, void > Struct Template Reference	206
	7.43.1	Detailed Description	207
	7.43.2	Constructor & Destructor Documentation	207
		7.43.2.1 Callback() [1/2]	207
		7.43.2.2 Callback() [2/2]	207
	7.43.3	Member Function Documentation	208
		7.43.3.1 execute()	208
		7.43.3.2 isValid()	208
7.44	Keyboa	ard::CallbackArea Struct Reference	208
7.45	Canvas	Class Reference	208
	7.45.1	Detailed Description	209
	7.45.2	Constructor & Destructor Documentation	209
		7.45.2.1 Canvas()	209
		7.45.2.2 ~Canvas()	209
	7.45.3	Member Function Documentation	209
		7.45.3.1 lineTo() [1/2]	210
		7.45.3.2 lineTo() [2/2]	210
		7.45.3.3 moveTo() [1/2]	210
		7.45.3.4 moveTo() [2/2]	211
		7.45.3.5 render()	211
7.46	Canvas	Widget Class Reference	211
	7.46.1	Detailed Description	212
	7.46.2	Constructor & Destructor Documentation	212
		7.46.2.1 CanvasWidget()	212
		7.46.2.2 ~CanvasWidget()	212
	7.46.3	Member Function Documentation	212
		7.46.3.1 draw()	213
		7.46.3.2 drawCanvasWidget()	213
		7.46.3.3 getAlpha()	213
		7.46.3.4 getMinimalRect()	214
		7.46.3.5 getPainter()	214
		7.46.3.6 getSolidRect()	214
		7.46.3.7 invalidate()	214
		7.46.3.8 setAlpha()	215
		7.46.3.9 setPainter()	215
7.47	Canvas	WidgetRenderer Class Reference	215
	7.47.1	Detailed Description	216

xviii CONTENTS

	7.47.2	Member Fu	unction Documentation	 . 216
		7.47.2.1	getMissingBufferSize()	 . 216
		7.47.2.2	getOutlineBuffer()	 . 216
		7.47.2.3	getOutlineBufferSize()	 . 217
		7.47.2.4	getScanlineCounts()	 . 217
		7.47.2.5	getScanlineCovers()	 . 217
		7.47.2.6	getScanlineStartIndices()	 . 217
		7.47.2.7	getScanlineWidth()	 . 217
		7.47.2.8	getUsedBufferSize()	 . 218
		7.47.2.9	getWriteMemoryUsageReport()	 . 218
		7.47.2.10 h	hasBuffer()	 . 218
		7.47.2.11 r	numCellsMissing()	 . 218
		7.47.2.12 r	numCellsUsed()	 . 219
		7.47.2.13	setScanlineWidth()	 . 219
		7.47.2.14	setupBuffer()	 . 219
		7.47.2.15	setWriteMemoryUsageReport()	 . 219
7.48	Cell Str	uct Referen	nce	 . 220
	7.48.1	Detailed De	escription	 . 221
	7.48.2	Member Fu	unction Documentation	 . 221
		7.48.2.1	addCover()	 . 221
		7.48.2.2 p	packedCoord()	 . 221
		7.48.2.3	set()	 . 221
		7.48.2.4	setCoord()	 . 222
		7.48.2.5	setCover()	 . 222
7.49	Circle C	Class Refere	ence	 . 222
	7.49.1	Detailed De	escription	 . 224
•	7.49.2	Constructo	or & Destructor Documentation	 . 224
		7.49.2.1	Circle()	 . 224
•	7.49.3	Member Fu	unction Documentation	 . 224
		7.49.3.1	drawCanvasWidget()	 . 224
		7.49.3.2	getArc()	 . 224
		7.49.3.3	getArcEnd() [1/2]	 . 225
		7.49.3.4	getArcEnd() [2/2]	 . 225
		7.49.3.5	getArcStart() [1/2]	 . 226
		7.49.3.6	getArcStart() [2/2]	 . 226
		7.49.3.7	getCapPrecision()	 . 226
		7.49.3.8	getCenter()	 . 227
		7.49.3.9	getLineWidth()	 . 227
		7.49.3.10 (getMinimalRect() [1/3]	 . 227
		7.49.3.11	getMinimalRect() [2/3]	 . 228

CONTENTS xix

		7.49.3.12 getMinimalRect() [3/3]	28
		7.49.3.13 getPrecision()	28
		7.49.3.14 getRadius()	29
		7.49.3.15 setArc() [1/2]	29
		7.49.3.16 setArc() [2/2]	29
		7.49.3.17 setCapPrecision()	30
		7.49.3.18 setCenter() [1/2]	30
		7.49.3.19 setCenter() [2/2]	31
		7.49.3.20 setCircle() [1/2]	31
		7.49.3.21 setCircle() [2/2]	31
		7.49.3.22 setLineWidth()	32
		7.49.3.23 setPrecision()	32
		7.49.3.24 setRadius()	33
		7.49.3.25 updateArc()	33
		7.49.3.26 updateArcEnd()	33
		7.49.3.27 updateArcStart()	35
7.50	CircleP	Progress Class Reference	35
	7.50.1	Detailed Description	36
	7.50.2	Constructor & Destructor Documentation	37
		7.50.2.1 CircleProgress()	37
		7.50.2.2 ~CircleProgress()	37
	7.50.3	Member Function Documentation	37
		7.50.3.1 getAlpha()	37
		7.50.3.2 getCapPrecision()	37
		7.50.3.3 getCenter()	38
		7.50.3.4 getEndAngle()	38
		7.50.3.5 getLineWidth()	38
		7.50.3.6 getRadius()	38
		7.50.3.7 getStartAngle()	39
		7.50.3.8 setAlpha()	39
		7.50.3.9 setCapPrecision()	39
		7.50.3.10 setCenter()	39
		7.50.3.11 setLineWidth()	40
		7.50.3.12 setPainter()	40
		7.50.3.13 setProgressIndicatorPosition()	40
		7.50.3.14 setRadius()	41
		7.50.3.15 setStartEndAngle()	41
		7.50.3.16 setValue()	41
7.51	ClickBu	uttonTrigger Class Reference	42
	7.51.1	Detailed Description	42

CONTENTS

	7.51.2	Member Function Documentation
		7.51.2.1 handleClickEvent()
7.52	ClickEv	vent Class Reference
	7.52.1	Detailed Description
	7.52.2	Member Enumeration Documentation
		7.52.2.1 ClickEventType
	7.52.3	Constructor & Destructor Documentation
		7.52.3.1 ClickEvent()
		7.52.3.2 ~ClickEvent()
	7.52.4	Member Function Documentation
		7.52.4.1 getEventType()
		7.52.4.2 getForce()
		7.52.4.3 getType()
		7.52.4.4 getX()
		7.52.4.5 getY()
		7.52.4.6 setType()
		7.52.4.7 setX()
		7.52.4.8 setY()
7.53	ClickLis	stener< T > Class Template Reference
	7.53.1	Detailed Description
	7.53.2	Constructor & Destructor Documentation
		7.53.2.1 ClickListener()
	7.53.3	Member Function Documentation
		7.53.3.1 handleClickEvent()
		7.53.3.2 setClickAction()
7.54	Color C	Class Reference
	7.54.1	Detailed Description
	7.54.2	Member Function Documentation
		7.54.2.1 getBlueColor()
		7.54.2.2 getColorFrom24BitRGB()
		7.54.2.3 getGreenColor()
		7.54.2.4 getRedColor()
7.55	colorty	pe Struct Reference
	7.55.1	Detailed Description
	7.55.2	Constructor & Destructor Documentation
		7.55.2.1 colortype() [1/2]
		7.55.2.2 colortype() [2/2]
	7.55.3	Member Function Documentation
		7.55.3.1 getColor32()
		7.55.3.2 operator uint16_t()

CONTENTS xxi

7.56	ConstF	ont Class Reference
	7.56.1	Detailed Description
	7.56.2	Constructor & Destructor Documentation
		7.56.2.1 ConstFont()
	7.56.3	Member Function Documentation
		7.56.3.1 find()
		7.56.3.2 getGlyph()
		7.56.3.3 getKerning()
		7.56.3.4 getPixelData()
7.57	Contair	ner Class Reference
	7.57.1	Detailed Description
	7.57.2	Constructor & Destructor Documentation
		7.57.2.1 Container()
		7.57.2.2 ~Container()
	7.57.3	Member Function Documentation
		7.57.3.1 add()
		7.57.3.2 contains()
		7.57.3.3 draw()
		7.57.3.4 forEachChild()
		7.57.3.5 getContainedArea()
		7.57.3.6 getFirstChild()
		7.57.3.7 getLastChild()
		7.57.3.8 getSolidRect()
		7.57.3.9 getType()
		7.57.3.10 insert()
		7.57.3.11 moveChildrenRelative()
		7.57.3.12 remove()
		7.57.3.13 removeAll()
		7.57.3.14 setupDrawChain()
		7.57.3.15 unlink()
7.58	CoverT	ransition < templateDirection > Class Template Reference
	7.58.1	Detailed Description
	7.58.2	Constructor & Destructor Documentation
		7.58.2.1 CoverTransition()
		7.58.2.2 ~CoverTransition()
	7.58.3	Member Function Documentation
		7.58.3.1 handleTickEvent()
		7.58.3.2 init()
		7.58.3.3 initMoveDrawable()
		7.58.3.4 tearDown()

xxii CONTENTS

		7.58.3.5 tickMoveDrawable()
7.59	CWRU	til Struct Reference
	7.59.1	Detailed Description
	7.59.2	Member Function Documentation
		7.59.2.1 angle() [1/4]
		7.59.2.2 angle() [2/4]
		7.59.2.3 angle() [3/4]
		7.59.2.4 angle() [4/4]
		7.59.2.5 arcsine()
		7.59.2.6 cosine() [1/2]
		7.59.2.7 cosine() [2/2]
		7.59.2.8 muldivQ10()
		7.59.2.9 muldivQ5()
		7.59.2.10 mulQ5() [1/2]
		7.59.2.11 mulQ5() [2/2]
		7.59.2.12 sine() [1/2]
		7.59.2.13 sine() [2/2]
		7.59.2.14 sqrtQ10()
		7.59.2.15 toQ10()
		7.59.2.16 toQ5() [1/2]
		7.59.2.17 toQ5() [2/2]
7.60	Debugl	Printer Class Reference
	7.60.1	Detailed Description
	7.60.2	Member Function Documentation
		7.60.2.1 draw()
		7.60.2.2 region()
		7.60.2.3 setDebugColor()
		7.60.2.4 setDebugPosition()
		7.60.2.5 setDebugScale()
		7.60.2.6 setDebugString()
7.61	Digital	Clock Class Reference
	7.61.1	Detailed Description
	7.61.2	Member Enumeration Documentation
		7.61.2.1 DisplayMode
	7.61.3	Constructor & Destructor Documentation
		7.61.3.1 DigitalClock()
		7.61.3.2 ~DigitalClock()
	7.61.4	Member Function Documentation
		7.61.4.1 displayLeadingZeroForHourIndicator()
		7.61.4.2 getAlpha()

CONTENTS xxiii

		7.61.4.3	getDisplayMode()	75
		7.61.4.4	getTextWidth()	75
		7.61.4.5	setAlpha()	75
		7.61.4.6	setBaselineY()	75
		7.61.4.7	setColor()	75
		7.61.4.8	setDisplayMode()	76
		7.61.4.9	setHeight()	76
		7.61.4.10	setTypedText()	76
		7.61.4.11	setWidth()	76
		7.61.4.12	updateClock()	77
7.62	Display	Transform	ation Class Reference	77
	7.62.1	Detailed I	Description	77
	7.62.2	Member I	Function Documentation	77
		7.62.2.1	transformDisplayToFrameBuffer() [1/6]	78
		7.62.2.2	transformDisplayToFrameBuffer() [2/6]	78
		7.62.2.3	transformDisplayToFrameBuffer() [3/6]	78
		7.62.2.4	transformDisplayToFrameBuffer() [4/6]	78
		7.62.2.5	transformDisplayToFrameBuffer() [5/6]	79
		7.62.2.6	transformDisplayToFrameBuffer() [6/6]	79
		7.62.2.7	transformFrameBufferToDisplay()	79
7.63	DMA_I	nterface C	lass Reference	79
	7.63.1	Detailed I	Description	81
	7.63.2	Construc	tor & Destructor Documentation	81
		7.63.2.1	~DMA_Interface()	81
		7.63.2.2	DMA_Interface()	81
	7.63.3	Member I	Function Documentation	81
		7.63.3.1	addToQueue()	81
		7.63.3.2	disableAlpha()	82
		7.63.3.3	enableAlpha()	82
		7.63.3.4	enableCopyWithTransparentPixels()	82
		7.63.3.5	execute()	82
		7.63.3.6	executeCompleted()	82
		7.63.3.7	flush()	82
		7.63.3.8	getAllowed()	82
		7.63.3.9	getBlitCaps()	83
		7.63.3.10	getDMAType()	83
		7.63.3.11	initialize()	83
		7.63.3.12	isDmaQueueEmpty()	83
		7.63.3.13	isDmaQueueFull()	84
		7.63.3.14	isDMARunning()	84

xxiv CONTENTS

		7.63.3.15 seedExecution()	34
		7.63.3.16 setAllowed()	34
		7.63.3.17 setupDataCopy()	34
		7.63.3.18 setupDataFill()	35
		7.63.3.19 signalDMAInterrupt()	35
		7.63.3.20 start()	35
		7.63.3.21 waitForFrameBufferSemaphore()	35
7.64	DMA_C	Queue Class Reference	35
	7.64.1	Detailed Description	36
	7.64.2	Constructor & Destructor Documentation	36
		7.64.2.1 ~DMA_Queue()	36
		7.64.2.2 DMA_Queue()	36
	7.64.3	Member Function Documentation	36
		7.64.3.1 first()	36
		7.64.3.2 isEmpty()	37
		7.64.3.3 isFull()	37
		7.64.3.4 pop()	37
		7.64.3.5 pushCopyOf()	37
7.65	DragEv	vent Class Reference	37
	7.65.1	Detailed Description	38
	7.65.2	Constructor & Destructor Documentation	38
		7.65.2.1 DragEvent()	38
		7.65.2.2 ~DragEvent()	39
	7.65.3	Member Function Documentation	39
		7.65.3.1 getDeltaX()	39
		7.65.3.2 getDeltaY()	39
		7.65.3.3 getEventType()	39
		7.65.3.4 getNewX()	90
		7.65.3.5 getNewY()	90
		7.65.3.6 getOldX()	90
		7.65.3.7 getOldY()	90
		7.65.3.8 getType()	90
7.66	Dragga	ble < T > Class Template Reference	91
	7.66.1	Detailed Description	91
	7.66.2	Constructor & Destructor Documentation	91
		7.66.2.1 Draggable()	91
		7.66.2.2 ~Draggable()	91
	7.66.3	Member Function Documentation	91
		7.66.3.1 handleDragEvent()	91
7.67	Drawab	ole Class Reference	92

CONTENTS xxv

7.67.1	Detailed Description
7.67.2	Member Enumeration Documentation
	7.67.2.1 DrawableType
7.67.3	Constructor & Destructor Documentation
	7.67.3.1 Drawable()
	7.67.3.2 ~Drawable()
7.67.4	Member Function Documentation
	7.67.4.1 childGeometryChanged()
	7.67.4.2 draw()
	7.67.4.3 drawToDynamicBitmap()
	7.67.4.4 getAbsoluteRect()
	7.67.4.5 getCachedAbsX()
	7.67.4.6 getCachedAbsY()
	7.67.4.7 getCachedVisibleRect()
	7.67.4.8 getHeight()
	7.67.4.9 getLastChild()
	7.67.4.10 getNextSibling()
	7.67.4.11 getParent()
	7.67.4.12 getRect()
	7.67.4.13 getSolidRect()
	7.67.4.14 getSolidRectAbsolute()
	7.67.4.15 getType()
	7.67.4.16 getVisibleRect()
	7.67.4.17 getWidth()
	7.67.4.18 getX()
	7.67.4.19 getY()
	7.67.4.20 handleClickEvent()
	7.67.4.21 handleDragEvent()
	7.67.4.22 handleGestureEvent()
	7.67.4.23 handleTickEvent()
	7.67.4.24 invalidate()
	7.67.4.25 invalidateRect()
	7.67.4.26 isTouchable()
	7.67.4.27 isVisible()
	7.67.4.28 moveRelative()
	7.67.4.29 moveTo()
	7.67.4.30 resetDrawChainCache()
	7.67.4.31 setHeight()
	7.67.4.32 setPosition()
	7.67.4.33 setTouchable()
	7.67.1.56 561.6661.456()

XXVI

	7.67.4.34 setupDrawChain()
	7.67.4.35 setVisible()
	7.67.4.36 setWidth()
	7.67.4.37 setX()
	7.67.4.38 setXY()
	7.67.4.39 setY()
	7.67.4.40 translateRectToAbsolute()
7.68 Drawa	bleList Class Reference
7.68.1	Detailed Description
7.68.2	Constructor & Destructor Documentation
	7.68.2.1 DrawableList()
	7.68.2.2 ~DrawableList()
7.68.3	Member Function Documentation
	7.68.3.1 getCircular()
	7.68.3.2 getDrawableIndex()
	7.68.3.3 getDrawableIndices()
	7.68.3.4 getDrawableMargin()
	7.68.3.5 getDrawableSize()
	7.68.3.6 getHorizontal()
	7.68.3.7 getItemIndex()
	7.68.3.8 getItemSize()
	7.68.3.9 getNumberOfDrawables()
	7.68.3.10 getNumberOfItems()
	7.68.3.11 getOffset()
	7.68.3.12 getRequiredNumberOfDrawables()
	7.68.3.13 itemChanged()
	7.68.3.14 refreshDrawables()
	7.68.3.15 setCircular()
	7.68.3.16 setDrawables()
	7.68.3.17 setDrawableSize()
	7.68.3.18 setHeight()
	7.68.3.19 setHorizontal()
	7.68.3.20 setNumberOfItems()
	7.68.3.21 setOffset()
	7.68.3.22 setWidth()
7.69 Drawa	bleListItems< TYPE, SIZE > Class Template Reference
7.69.1	Detailed Description
7.69.2	Constructor & Destructor Documentation
	7.69.2.1 DrawableListItems()
	7.69.2.2 ~DrawableListItems()

CONTENTS xxvii

	7.69.3	Member Function Documentation	316
		7.69.3.1 getDrawable()	316
		7.69.3.2 getNumberOfDrawables()	317
		7.69.3.3 operator[]()	317
7.70	Drawak	oleListItemsInterface Class Reference	317
	7.70.1	Detailed Description	317
	7.70.2	Constructor & Destructor Documentation	318
		7.70.2.1 ~DrawableListItemsInterface()	318
	7.70.3	Member Function Documentation	318
		7.70.3.1 getDrawable()	318
		7.70.3.2 getNumberOfDrawables()	318
7.71	Drawin	gSurface Struct Reference	318
7.72	Bitmap	::DynamicBitmapData Struct Reference	319
	7.72.1	Detailed Description	319
7.73	Easing	Equations Class Reference	319
	7.73.1	Detailed Description	321
	7.73.2	Member Function Documentation	321
		7.73.2.1 backEaseIn()	321
		7.73.2.2 backEaseInOut()	321
		7.73.2.3 backEaseOut()	322
		7.73.2.4 bounceEaseIn()	322
		7.73.2.5 bounceEaseInOut()	323
		7.73.2.6 bounceEaseOut()	323
		7.73.2.7 circEaseIn()	323
		7.73.2.8 circEaseInOut()	324
		7.73.2.9 circEaseOut()	324
		7.73.2.10 cubicEaseIn()	325
		7.73.2.11 cubicEaseInOut()	325
		7.73.2.12 cubicEaseOut()	326
		7.73.2.13 elasticEaseIn()	326
		7.73.2.14 elasticEaseInOut()	326
		7.73.2.15 elasticEaseOut()	327
		7.73.2.16 expoEaseIn()	327
		7.73.2.17 expoEaseInOut()	328
		7.73.2.18 expoEaseOut()	328
		7.73.2.19 linearEaseIn()	329
		7.73.2.20 linearEaseInOut()	329
		7.73.2.21 linearEaseNone()	329
		7.73.2.22 linearEaseOut()	330
		7.73.2.23 quadEaseIn()	330

xxviii CONTENTS

	7.73.2.24	4 quadEaseInOut()	. 331
	7.73.2.25	5 quadEaseOut()	. 331
	7.73.2.26	6 quartEaseIn()	. 331
	7.73.2.27	7 quartEaseInOut()	. 332
	7.73.2.28	B quartEaseOut()	. 332
	7.73.2.29	9 quintEaseIn()	. 333
	7.73.2.30	O quintEaseInOut()	. 333
	7.73.2.31	1 quintEaseOut()	. 334
	7.73.2.32	2 sineEaseIn()	. 334
	7.73.2.33	3 sineEaseInOut()	. 334
	7.73.2.34	4 sineEaseOut()	. 335
7.74 Edge	Struct Refe	erence	. 335
7.74.1	Construc	ctor & Destructor Documentation	. 336
	7.74.1.1	Edge()	. 336
7.74.2	Member	Function Documentation	. 337
	7.74.2.1	step() [1/2]	. 337
	7.74.2.2	step() [2/2]	. 337
7.75 Event	Class Refe	erence	. 337
7.75.1	Detailed	Description	. 338
7.75.2	Member l	Enumeration Documentation	. 338
	7.75.2.1	EventType	. 338
7.75.3	3 Construc	ctor & Destructor Documentation	. 338
	7.75.3.1	~Event()	. 338
7.75.4	Member	Function Documentation	. 338
	7.75.4.1	getEventType()	. 338
7.76 FadeA	Animator<	T > Class Template Reference	. 339
7.76.1	Detailed	Description	. 340
7.76.2	2 Construc	ctor & Destructor Documentation	. 340
	7.76.2.1	FadeAnimator()	. 340
	7.76.2.2	~FadeAnimator()	. 340
7.76.3	Member	Function Documentation	. 340
	7.76.3.1	clearFadeAnimationEndedAction()	. 340
	7.76.3.2	getFadeAnimationDelay()	. 341
	7.76.3.3	handleTickEvent()	. 341
	7.76.3.4	isFadeAnimationRunning()	. 341
	7.76.3.5	isRunning()	. 341
	7.76.3.6	nextFadeAnimationStep()	. 341
	7.76.3.7	setFadeAnimationDelay()	. 341
	7.76.3.8	setFadeAnimationEndedAction()	. 342
	7.76.3.9	startFadeAnimation()	. 342

CONTENTS xxix

7.77 FlashDataReader Class Reference	342
7.77.1 Detailed Description	343
7.77.2 Member Function Documentation	343
7.77.2.1 addressIsAddressable()	343
7.77.2.2 copyData()	343
7.77.2.3 startFlashLineRead()	343
7.77.2.4 waitFlashReadComplete()	344
7.78 Font Class Reference	344
7.78.1 Detailed Description	346
7.78.2 Constructor & Destructor Documentation	346
7.78.2.1 Font()	346
7.78.3 Member Function Documentation	346
7.78.3.1 getBitsPerPixel()	346
7.78.3.2 getCharWidth()	347
7.78.3.3 getDataFormatA4()	347
7.78.3.4 getEllipsisChar()	347
7.78.3.5 getFallbackChar()	347
7.78.3.6 getFontHeight()	348
7.78.3.7 getGlyph() [1/2]	348
7.78.3.8 getGlyph() [2/2]	348
7.78.3.9 getGSUBTable()	349
7.78.3.10 getKerning()	349
7.78.3.11 getMaxPixelsLeft()	349
7.78.3.12 getMaxPixelsRight()	350
7.78.3.13 getMaxTextHeight()	350
7.78.3.14 getMinimumTextHeight()	
7.78.3.15 getNumberOfLines()	
7.78.3.16 getSpacingAbove()	
7.78.3.17 getStringWidth() [1/2]	
7.78.3.18 getStringWidth() [2/2]	
7.78.3.19 getStringWidthLTR()	
7.78.3.20 getStringWidthRTL()	
7.79 FontManager Class Reference	
7.79.1 Detailed Description	
7.79.2 Member Function Documentation	
7.79.2.1 getFont()	
7.79.2.2 setFontProvider()	
7.80 FontProvider Class Reference	
7.80.1 Detailed Description	
7.80.2 Constructor & Destructor Documentation	

CONTENTS

		7.80.2.1 ~FontProvider()	354
	7.80.3	Member Function Documentation	354
		7.80.3.1 getFont()	354
7.81	FrameE	BufferAllocator Class Reference	355
	7.81.1	Detailed Description	355
	7.81.2	Member Function Documentation	355
		7.81.2.1 allocateBlock()	355
		7.81.2.2 freeBlockAfterTransfer()	356
		7.81.2.3 getBlockForTransfer()	356
		7.81.2.4 hasBlockReadyForTransfer()	356
		7.81.2.5 markBlockReadyForTransfer()	357
7.82	CoverT	ransition< templateDirection >::FullSolidRect Class Reference	357
	7.82.1	Member Function Documentation	357
		7.82.1.1 draw()	357
		7.82.1.2 getSolidRect()	357
7.83	Generi	cCallback< T1, T2, T3 > Class Template Reference	358
	7.83.1	Detailed Description	358
	7.83.2	Constructor & Destructor Documentation	358
		7.83.2.1 ~GenericCallback()	359
	7.83.3	Member Function Documentation	359
		7.83.3.1 execute()	359
		7.83.3.2 isValid()	359
7.84	Generi	cCallback< T1, T2, void > Class Template Reference	359
	7.84.1	Detailed Description	360
	7.84.2	Constructor & Destructor Documentation	360
		7.84.2.1 ∼GenericCallback()	360
	7.84.3	Member Function Documentation	360
		7.84.3.1 execute()	361
		7.84.3.2 isValid()	361
7.85	Generi	cCallback< T1, void, void > Class Template Reference	361
	7.85.1	Detailed Description	361
	7.85.2	Constructor & Destructor Documentation	362
		7.85.2.1 ∼GenericCallback()	362
	7.85.3	Member Function Documentation	362
		7.85.3.1 execute()	362
		7.85.3.2 isValid()	362
7.86	Generi	cCallback< void > Class Template Reference	362
	7.86.1	Detailed Description	363
	7.86.2	Constructor & Destructor Documentation	363
		7.86.2.1 ~GenericCallback()	363

CONTENTS xxxi

	7.86.3	Member Function Documentation	33
		7.86.3.1 execute()	33
		7.86.3.2 isValid()	34
7.87	Gesture	eEvent Class Reference	34
	7.87.1	Detailed Description	34
	7.87.2	Member Enumeration Documentation	34
		7.87.2.1 GestureType	3 4
	7.87.3	Constructor & Destructor Documentation	
		7.87.3.1 GestureEvent()	
	7.87.4	Member Function Documentation	
		7.87.4.1 getEventType()	35
		7.87.4.2 getType()	35
		7.87.4.3 getVelocity()	36
		7.87.4.4 getX()	36
		7.87.4.5 getY()	36
7.88	Gesture	es Class Reference	36
	7.88.1	Detailed Description	37
	7.88.2	Constructor & Destructor Documentation	37
		7.88.2.1 Gestures()	37
	7.88.3	Member Function Documentation	37
		7.88.3.1 registerClickEvent()	37
		7.88.3.2 registerDragEvent()	37
		7.88.3.3 registerEventListener()	86
		7.88.3.4 setDragThreshold()	38
		7.88.3.5 tick()	38
7.89	GlyphN	Node Struct Reference	38
	7.89.1	Detailed Description	39
	7.89.2	Member Function Documentation	39
		7.89.2.1 advance()	39
		7.89.2.2 height()	70
		7.89.2.3 kerningTablePos()	70
		7.89.2.4 setTop()	70
		7.89.2.5 top()	70
		7.89.2.6 width()	70
7.90	GPIO (Class Reference	71
	7.90.1	Detailed Description	71
	7.90.2	Member Enumeration Documentation	71
		7.90.2.1 GPIO_ID	71
	7.90.3	Member Function Documentation	71
		7.90.3.1 clear()	72

xxxii CONTENTS

	7.90.3.2 get()
	7.90.3.3 init()
	7.90.3.4 set()
	7.90.3.5 toggle()
7.91 Gradie	nts Struct Reference
7.91.1	Constructor & Destructor Documentation
	7.91.1.1 Gradients()
7.92 HAL C	lass Reference
7.92.1	Detailed Description
7.92.2	Member Enumeration Documentation
	7.92.2.1 FrameRefreshStrategy
7.92.3	Constructor & Destructor Documentation
	7.92.3.1 HAL()
	7.92.3.2 ~HAL()
7.92.4	Member Function Documentation
	7.92.4.1 allowDMATransfers()
	7.92.4.2 backPorchExited()
	7.92.4.3 beginFrame()
	7.92.4.4 blitCopy() [1/3]
	7.92.4.5 blitCopy() [2/3]
	7.92.4.6 blitCopy() [3/3]
	7.92.4.7 blitCopyARGB8888()
	7.92.4.8 blitCopyGlyph()
	7.92.4.9 blitFill() [1/2]
	7.92.4.10 blitFill() [2/2]
	7.92.4.11 blitSetTransparencyKey()
	7.92.4.12 blockCopy()
	7.92.4.13 cacheTextString()
	7.92.4.14 configureInterrupts()
	7.92.4.15 configurePartialFrameBuffer()
	7.92.4.16 copyFBRegionToMemory() [1/2]
	7.92.4.17 copyFBRegionToMemory() [2/2]
	7.92.4.18 disableInterrupts()
	7.92.4.19 drawDrawableInDynamicBitmap() [1/2]
	7.92.4.20 drawDrawableInDynamicBitmap() [2/2]
	7.92.4.21 enableInterrupts()
	7.92.4.22 enableLCDControllerInterrupt()
	7.92.4.23 enableMCULoadCalculation()
	7.92.4.24 endFrame()
	7.92.4.25 flushDMA()

CONTENTS xxxiii

7.92.4.26 flushFrameBuffer() [1/2]
7.92.4.27 flushFrameBuffer() [2/2]
7.92.4.28 frontPorchEntered()
7.92.4.29 getAnimationStorage()
7.92.4.30 getAuxiliaryLCD()
7.92.4.31 getBlitCaps()
7.92.4.32 getButtonController()
7.92.4.33 getClientFrameBuffer()
7.92.4.34 getCPUCycles()
7.92.4.35 getDisplayHeight()
7.92.4.36 getDisplayOrientation()
7.92.4.37 getDisplayWidth()
7.92.4.38 getDMAType()
7.92.4.39 getFingerSize()
7.92.4.40 getFrameBufferAllocator()
7.92.4.41 getFrameRefreshStrategy()
7.92.4.42 getInstance()
7.92.4.43 getLCDRefreshCount()
7.92.4.44 getMCULoadPct()
7.92.4.45 getTFTCurrentLine()
7.92.4.46 getTFTFrameBuffer()
7.92.4.47 getTouchSampleRate()
7.92.4.48 initialize()
7.92.4.49 lcd()
7.92.4.50 lockDMAToFrontPorch()
7.92.4.51 lockFrameBuffer()
7.92.4.52 noTouch()
7.92.4.53 performDisplayOrientationChange()
7.92.4.54 registerEventListener()
7.92.4.55 registerTaskDelayFunction()
7.92.4.56 registerTextCache()
7.92.4.57 sampleKey()
7.92.4.58 setAuxiliaryLCD()
7.92.4.59 setButtonController()
7.92.4.60 setDisplayOrientation()
7.92.4.61 setDragThreshold()
7.92.4.62 setFingerSize()
7.92.4.63 setFrameBufferAllocator()
7.92.4.64 setFrameBufferStartAddress()
7.92.4.65 setFrameBufferStartAddresses()

CONTENTS

	7.92.4.66 setFrameRateCompensation()	398
	7.92.4.67 setFrameRefreshStrategy()	399
	7.92.4.68 setMCUActive()	399
	7.92.4.69 setMCUInstrumentation()	399
	7.92.4.70 setTFTFrameBuffer()	400
	7.92.4.71 setTouchSampleRate()	400
	7.92.4.72 signalDMAInterrupt()	400
	7.92.4.73 swapFrameBuffers()	400
	7.92.4.74 taskDelay()	400
	7.92.4.75 taskEntry()	401
	7.92.4.76 tick()	401
	7.92.4.77 touch()	401
	7.92.4.78 unlockFrameBuffer()	401
	7.92.4.79 vSync()	401
7.93 HALSE	DL2 Class Reference	402
7.93.1	Detailed Description	403
7.93.2	Constructor & Destructor Documentation	403
	7.93.2.1 HALSDL2()	403
7.93.3	Member Function Documentation	404
	7.93.3.1 blitSetTransparencyKey()	404
	7.93.3.2 blockCopy()	404
	7.93.3.3 configureInterrupts()	404
	7.93.3.4 configureLCDInterrupt()	404
	7.93.3.5 copyScreenshotToClipboard()	405
	7.93.3.6 disableInterrupts()	405
	7.93.3.7 doRotate()	405
	7.93.3.8 doSampleTouch()	405
	7.93.3.9 enableInterrupts()	405
	7.93.3.10 enableLCDControllerInterrupt()	406
	7.93.3.11 flushFrameBuffer() [1/2]	406
	7.93.3.12 flushFrameBuffer() [2/2]	406
	7.93.3.13 getArgv()	406
	7.93.3.14 getTFTFrameBuffer()	407
	7.93.3.15 getWindowTitle()	407
	7.93.3.16 loadSkin()	407
	7.93.3.17 performDisplayOrientationChange()	408
	7.93.3.18 renderLCD_FrameBufferToMemory()	408
	7.93.3.19 sampleKey()	408
	7.93.3.20 saveNextScreenshots()	408
	7.93.3.21 saveScreenshot() [1/2]	409

CONTENTS XXXV

		7.93.3.22 saveScreenshot() [2/2]
		7.93.3.23 scaleTo24bpp()
		7.93.3.24 sdl_init()
		7.93.3.25 setTFTFrameBuffer()
		7.93.3.26 setVsyncInterval()
		7.93.3.27 setWindowTitle()
		7.93.3.28 taskEntry()
7.94	I2C Cla	ass Reference
	7.94.1	Detailed Description
	7.94.2	Constructor & Destructor Documentation
		7.94.2.1 I2C()
		7.94.2.2 ~I2C()
	7.94.3	Member Function Documentation
		7.94.3.1 init()
		7.94.3.2 readRegister()
		7.94.3.3 writeRegister()
7.95	I2CTou	chController Class Reference
	7.95.1	Detailed Description
	7.95.2	Constructor & Destructor Documentation
		7.95.2.1 I2CTouchController()
		7.95.2.2 ~I2CTouchController()
	7.95.3	Member Function Documentation
		7.95.3.1 init()
		7.95.3.2 sampleTouch()
7.96	IconBu	ttonStyle < T > Class Template Reference
	7.96.1	Detailed Description
	7.96.2	Member Function Documentation
		7.96.2.1 getCurrentlyDisplayedIcon()
		7.96.2.2 getlconX()
		7.96.2.3 getlconY()
		7.96.2.4 setIconBitmaps()
		7.96.2.5 setlconX()
		7.96.2.6 setlconXY()
		7.96.2.7 setIconY()
7.97	Image	Class Reference
	7.97.1	Detailed Description
	7.97.2	Constructor & Destructor Documentation
		7.97.2.1 Image()
	7.97.3	Member Function Documentation
		7.97.3.1 draw()

xxxvi CONTENTS

	7.97.3.2	getAlpha()
	7.97.3.3	getBitmap()
	7.97.3.4	getSolidRect()
	7.97.3.5	getType()
	7.97.3.6	setAlpha()
	7.97.3.7	setBitmap()
7.98 Image	ButtonStyle	e < T > Class Template Reference
7.98.1	Detailed	Description
7.98.2	Member	Function Documentation
	7.98.2.1	getCurrentlyDisplayedBitmap()
	7.98.2.2	setBitmaps()
	7.98.2.3	setBitmapXY()
7.99 Image	Progress C	Class Reference
7.99.1	Detailed	Description
7.99.2	Construc	ctor & Destructor Documentation
	7.99.2.1	ImageProgress()
	7.99.2.2	~ImageProgress()
7.99.3	Member	Function Documentation
	7.99.3.1	getAlpha()
	7.99.3.2	getAnchorAtZero()
	7.99.3.3	getBitmap()
	7.99.3.4	setAlpha()
	7.99.3.5	setAnchorAtZero()
	7.99.3.6	setBitmap()
	7.99.3.7	setProgressIndicatorPosition()
	7.99.3.8	setValue()
7.100Interna	alFlashFon	t Class Reference
7.100.	1 Detailed	Description
7.100.	2 Construc	stor & Destructor Documentation
	7.100.2.1	InternalFlashFont()
7.100.		Function Documentation
	7.100.3.1	getKerning()
		2 getPixelData()
7.101Scanlii		Class Reference
		Description
		stor & Destructor Documentation
		l iterator()
7.101		Function Documentation
		I getCovers()
		2 getNumPix()
	2	•

CONTENTS xxxvii

7.101.3.3 next()	429
7.102JSMOCHelper Class Reference	429
7.102.1 Detailed Description	430
7.102.2 Constructor & Destructor Documentation	430
7.102.2.1 JSMOCHelper()	430
7.102.3 Member Function Documentation	430
7.102.3.1 draw()	430
7.102.3.2 getCachedAbsX()	431
7.102.3.3 getCachedAbsY()	431
7.102.3.4 getCachedVisibleRect()	431
7.102.3.5 getHeight()	431
7.102.3.6 getWidget()	431
7.102.3.7 getWidth()	432
7.102.3.8 setWidget()	432
7.103KerningNode Struct Reference	432
7.103.1 Detailed Description	432
7.104Keyboard::Key Struct Reference	432
7.105Keyboard Class Reference	433
7.105.1 Detailed Description	434
7.105.2 Constructor & Destructor Documentation	435
7.105.2.1 Keyboard()	435
7.105.2.2 ~Keyboard()	435
7.105.3 Member Function Documentation	435
7.105.3.1 draw()	435
7.105.3.2 getBuffer()	435
7.105.3.3 getBufferPosition()	436
7.105.3.4 getCallbackAreaForCoordinates()	436
7.105.3.5 getCharForKey()	436
7.105.3.6 getKeyForCoordinates()	437
7.105.3.7 getKeyMappingList()	437
7.105.3.8 getLayout()	437
7.105.3.9 getType()	437
7.105.3.10handleClickEvent()	438
7.105.3.11handleDragEvent()	438
7.105.3.12setBuffer()	438
7.105.3.13setBufferPosition()	438
7.105.3.14setKeyListener()	439
7.105.3.15setKeymappingList()	439
7.105.3.16setLayout()	439
7.105.3.17setTextIndentation()	439

xxxviii CONTENTS

CONTENTS xxxix

7.109.3.32stringWidth()	457
7.110LCD16bpp Class Reference	457
7.110.1 Detailed Description	459
7.110.2 Member Function Documentation	459
7.110.2.1 bitDepth()	459
7.110.2.2 blitCopy() [1/2]	460
7.110.2.3 blitCopy() [2/2]	460
7.110.2.4 blitCopyAlphaPerPixel()	461
7.110.2.5 blitCopyARGB8888()	461
7.110.2.6 blitCopyL8()	461
7.110.2.7 blitCopyL8_ARGB8888()	462
7.110.2.8 blitCopyL8_RGB565()	462
7.110.2.9 blitCopyL8_RGB888()	463
7.110.2.10copyFrameBufferRegionToMemory()	463
7.110.2.11drawGlyph()	464
7.110.2.12drawPartialBitmap()	465
7.110.2.13drawTextureMapScanLine()	465
7.110.2.14fillRect()	466
7.110.2.15framebufferFormat()	466
7.110.2.16framebufferStride()	466
7.110.2.17getBlueColor()	466
7.110.2.18getBlueFromColor()	467
7.110.2.19getColorFrom24BitRGB()	467
7.110.2.20getColorFromRGB()	467
7.110.2.21getFramebufferStride()	468
7.110.2.22getGreenColor()	468
7.110.2.23getGreenFromColor()	468
7.110.2.24getRedColor()	469
7.110.2.25getRedFromColor()	469
7.110.2.26nit()	469
7.110.2.27nextLine()	469
7.110.2.28nextPixel()	470
7.111LCD16bppSerialFlash Class Reference	470
7.111.1 Detailed Description	472
7.111.2 Constructor & Destructor Documentation	472
7.111.2.1 LCD16bppSerialFlash()	472
7.111.3 Member Function Documentation	473
7.111.3.1 bitDepth()	473
7.111.3.2 blitCopy() [1/2]	473
7.111.3.3 blitCopy() [2/2]	473

xI CONTENTS

7.111.3.4 blitCopyARGB8888()	74
7.111.3.5 blitCopyL8()	74
7.111.3.6 blitCopyL8_ARGB8888()	75
7.111.3.7 blitCopyL8_RGB565()	75
7.111.3.8 copyFrameBufferRegionToMemory()	75
7.111.3.9 drawGlyph()	⁷ 6
7.111.3.10drawPartialBitmap()	77
7.111.3.11drawTextureMapScanLine()	77
7.111.3.12fillRect()	78
7.111.3.13framebufferFormat()	78
7.111.3.14framebufferStride()	79
7.111.3.15getBlueColor()	79
7.111.3.16getBlueFromColor()	79
7.111.3.17getColorFrom24BitRGB()	79
7.111.3.1&getColorFromRGB()	30
7.111.3.19getFramebufferStride()	30
7.111.3.20getGreenColor()	30
7.111.3.21getGreenFromColor()	31
7.111.3.22getRedColor()	31
7.111.3.23getRedFromColor()	31
7.111.3.24init()	32
7.111.3.25nextLine()	32
7.111.3.2@nextPixel()	32
7.112LCD1bpp Class Reference	32
7.112.1 Detailed Description	34
7.112.2 Member Function Documentation	34
7.112.2.1 bitDepth()	34
7.112.2.2 blitCopy() [1/2]	35
7.112.2.3 blitCopy() [2/2]	35
7.112.2.4 blitCopyRLE()	35
7.112.2.5 copyFrameBufferRegionToMemory()	36
7.112.2.6 copyRect()	36
7.112.2.7 drawGlyph()	37
7.112.2.8 drawPartialBitmap()	38
7.112.2.9 drawTextureMapScanLine()	38
7.112.2.10fillMemory()	39
7.112.2.1 ffillRect()	39
7.112.2.12framebufferFormat()	39
7.112.2.13framebufferStride()	90
7.112.2.14getBlueColor()	90

CONTENTS xli

7.112.2.15getBlueFromColor()	490
7.112.2.16getColorFrom24BitRGB()	491
7.112.2.17getColorFromRGB()	491
7.112.2.1&getFramebufferStride()	491
7.112.2.19getGreenColor()	492
7.112.2.20getGreenFromColor()	492
7.112.2.21getRedColor()	492
7.112.2.22getRedFromColor()	493
7.112.2.23nextLine()	493
7.112.2.24nextPixel()	493
7.113LCD24bpp Class Reference	493
7.113.1 Detailed Description	495
7.113.2 Member Function Documentation	496
7.113.2.1 bitDepth()	496
7.113.2.2 blitCopy() [1/2]	496
7.113.2.3 blitCopy() [2/2]	496
7.113.2.4 blitCopyARGB8888()	497
7.113.2.5 blitCopyL8()	497
7.113.2.6 blitCopyL8_ARGB8888()	498
7.113.2.7 blitCopyL8_RGB888()	498
7.113.2.8 copyFrameBufferRegionToMemory()	498
7.113.2.9 drawGlyph()	499
7.113.2.10drawPartialBitmap()	500
7.113.2.11drawTextureMapScanLine()	500
7.113.2.12fillRect()	501
7.113.2.13 ramebufferFormat()	. 501
7.113.2.14framebufferStride()	502
7.113.2.15getBlueColor()	502
7.113.2.16getBlueFromColor()	502
7.113.2.17getColorFrom24BitRGB()	502
7.113.2.18getColorFromRGB()	503
7.113.2.19getFramebufferStride()	503
7.113.2.20getGreenColor()	503
7.113.2.21getGreenFromColor()	504
7.113.2.22getRedColor()	504
7.113.2.23getRedFromColor()	504
7.113.2.24nit()	505
7.113.2.25nextLine()	505
7.113.2.26nextPixel()	505
7.114LCD24DebugPrinter Class Reference	505

XIII CONTENTS

7.114.1 Detailed Description	506
7.114.2 Member Function Documentation	506
7.114.2.1 draw()	506
7.115LCD2bpp Class Reference	506
7.115.1 Detailed Description	508
7.115.2 Member Function Documentation	508
7.115.2.1 bitDepth()	508
7.115.2.2 blitCopy() [1/2]	508
7.115.2.3 blitCopy() [2/2]	509
7.115.2.4 blitCopyAlphaPerPixel()	509
7.115.2.5 copyFrameBufferRegionToMemory()	510
7.115.2.6 copyRect()	510
7.115.2.7 drawGlyph()	511
7.115.2.8 drawPartialBitmap()	512
7.115.2.9 drawTextureMapScanLine()	512
7.115.2.10fillRect()	513
7.115.2.11framebufferFormat()	513
7.115.2.12framebufferStride()	513
7.115.2.13getBlueColor()	514
7.115.2.14getBlueFromColor()	514
7.115.2.15getColorFrom24BitRGB()	514
7.115.2.16getColorFromRGB()	515
7.115.2.17getFramebufferStride()	515
7.115.2.1&getGreenColor()	515
7.115.2.19getGreenFromColor()	516
7.115.2.20getRedColor()	516
7.115.2.21getRedFromColor()	516
7.115.2.2월nit()	516
7.115.2.23nextLine()	517
7.115.2.24nextPixel()	517
7.116LCD32bpp Class Reference	517
7.116.1 Detailed Description	519
7.116.2 Member Function Documentation	519
7.116.2.1 bitDepth()	519
7.116.2.2 blitCopy() [1/2]	520
7.116.2.3 blitCopy() [2/2]	520
7.116.2.4 blitCopyL8()	521
7.116.2.5 blitCopyL8_ARGB8888()	521
7.116.2.6 blitCopyL8_RGB565()	521
7.116.2.7 blitCopyL8_RGB888()	522

CONTENTS xliii

7.116.2.8 biitCopyRGB565()	522
7.116.2.9 blitCopyRGB888()	523
7.116.2.10copyFrameBufferRegionToMemory()	523
7.116.2.11drawGlyph()	524
7.116.2.12drawPartialBitmap()	524
7.116.2.13drawTextureMapScanLine()	525
7.116.2.14fillRect()	526
7.116.2.15framebufferFormat()	526
7.116.2.16framebufferStride()	526
7.116.2.17getBlueColor()	526
7.116.2.1&getBlueFromColor()	527
7.116.2.19getColorFrom24BitRGB()	527
7.116.2.20getColorFromRGB()	527
7.116.2.21getFramebufferStride()	528
7.116.2.22getGreenColor()	528
7.116.2.23getGreenFromColor()	528
7.116.2.24getRedColor()	529
7.116.2.25getRedFromColor()	529
7.116.2.26nit()	529
7.116.2.27nextLine()	529
7.116.2.28nextPixel()	530
7.117LCD4bpp Class Reference	530
7.117.1 Detailed Description	532
7.117.2 Member Function Documentation	532
7.117.2.1 bitDepth()	532
7.117.2.2 blitCopy() [1/2]	532
7.117.2.3 blitCopy() [2/2]	533
7.117.2.4 blitCopyAlphaPerPixel()	533
7.117.2.5 copyFrameBufferRegionToMemory()	534
7.117.2.6 copyRect()	534
7.117.2.7 drawGlyph()	535
7.117.2.8 drawPartialBitmap()	535
7.117.2.9 drawTextureMapScanLine()	536
7.117.2.10fillRect()	536
7.117.2.11framebufferFormat()	537
7.117.2.12framebufferStride()	537
7.117.2.13getBlueColor()	537
7.117.2.14getBlueFromColor()	538
7.117.2.15getColorFrom24BitRGB()	538
7.117.2.16getColorFromRGB()	538

XIIV CONTENTS

7.117.2.17getFramebufferStride()	539
7.117.2.18getGreenColor()	539
7.117.2.19getGreenFromColor()	539
7.117.2.20getRedColor()	539
7.117.2.21getRedFromColor()	540
7.117.2.22nit()	540
7.117.2.23nextLine()	540
7.117.2.24nextPixel()	541
7.118LCD8bpp_ABGR2222 Class Reference	541
7.118.1 Detailed Description	543
7.118.2 Member Function Documentation	543
7.118.2.1 bitDepth()	543
7.118.2.2 blitCopy() [1/2]	543
7.118.2.3 blitCopy() [2/2]	544
7.118.2.4 blitCopyAlphaPerPixel()	544
7.118.2.5 blitCopyARGB8888()	545
7.118.2.6 copyFrameBufferRegionToMemory()	545
7.118.2.7 drawGlyph()	546
7.118.2.8 drawPartialBitmap()	546
7.118.2.9 drawTextureMapScanLine()	547
7.118.2.10fillRect()	547
7.118.2.11framebufferFormat()	548
7.118.2.12framebufferStride()	548
7.118.2.13getBlueColor()	548
7.118.2.14getBlueFromColor()	549
7.118.2.15getColorFrom24BitRGB()	549
7.118.2.16getColorFromRGB()	549
7.118.2.17getFramebufferStride()	550
7.118.2.18getGreenColor()	550
7.118.2.19getGreenFromColor()	550
7.118.2.20getRedColor()	550
7.118.2.21getRedFromColor()	551
7.118.2.22nit()	551
7.118.2.23nextLine()	551
7.118.2.24nextPixel()	551
7.119LCD8bpp_ARGB2222 Class Reference	552
7.119.1 Detailed Description	553
7.119.2 Member Function Documentation	554
7.119.2.1 bitDepth()	
7.119.2.2 blitCopy() [1/2]	554

CONTENTS xiv

7.119.2.3 blitCopy() [2/2]	 554
7.119.2.4 blitCopyAlphaPerPixel()	 555
7.119.2.5 blitCopyARGB8888()	 555
7.119.2.6 copyFrameBufferRegionToMemory()	 556
7.119.2.7 drawGlyph()	 556
7.119.2.8 drawPartialBitmap()	 557
7.119.2.9 drawTextureMapScanLine()	 558
7.119.2.10fillRect()	 558
7.119.2.11framebufferFormat()	 559
7.119.2.12framebufferStride()	 559
7.119.2.13getBlueColor()	 559
7.119.2.14getBlueFromColor()	 559
7.119.2.15getColorFrom24BitRGB()	 560
7.119.2.16getColorFromRGB()	 560
7.119.2.17getFramebufferStride()	 560
7.119.2.1&getGreenColor()	 561
7.119.2.19getGreenFromColor()	 561
7.119.2.20getRedColor()	 561
7.119.2.21getRedFromColor()	 562
7.119.2.22nit()	 562
7.119.2.23nextLine()	 562
7.119.2.24nextPixel()	 562
7.120LCD8bpp_BGRA2222 Class Reference	 563
7.120.1 Detailed Description	 564
7.120.2 Member Function Documentation	 565
7.120.2.1 bitDepth()	 565
7.120.2.2 blitCopy() [1/2]	 565
7.120.2.3 blitCopy() [2/2]	 565
7.120.2.4 blitCopyAlphaPerPixel()	 566
7.120.2.5 blitCopyARGB8888()	 566
7.120.2.6 copyFrameBufferRegionToMemory()	 567
7.120.2.7 drawGlyph()	 567
7.120.2.8 drawPartialBitmap()	 568
7.120.2.9 drawTextureMapScanLine()	 569
7.120.2.10fillRect()	 569
7.120.2.11framebufferFormat()	 570
7.120.2.12framebufferStride()	 570
7.120.2.13getBlueColor()	 570
7.120.2.14getBlueFromColor()	 570
7.120.2.15getColorFrom24BitRGB()	 571

XIVI

7.120.2.16getColorFromRGB()	. 571
7.120.2.17getFramebufferStride()	. 571
7.120.2.1&getGreenColor()	. 572
7.120.2.19getGreenFromColor()	. 572
7.120.2.20getRedColor()	. 572
7.120.2.21getRedFromColor()	. 573
7.120.2.2월nit()	. 573
7.120.2.23nextLine()	. 573
7.120.2.24nextPixel()	. 573
7.121LCD8bpp_RGBA2222 Class Reference	. 574
7.121.1 Detailed Description	. 575
7.121.2 Member Function Documentation	. 576
7.121.2.1 bitDepth()	. 576
7.121.2.2 blitCopy() [1/2]	. 576
7.121.2.3 blitCopy() [2/2]	. 576
7.121.2.4 blitCopyAlphaPerPixel()	. 577
7.121.2.5 blitCopyARGB8888()	. 577
7.121.2.6 copyFrameBufferRegionToMemory()	. 578
7.121.2.7 drawGlyph()	. 578
7.121.2.8 drawPartialBitmap()	. 579
7.121.2.9 drawTextureMapScanLine()	. 580
7.121.2.10fillRect()	. 580
7.121.2.11framebufferFormat()	. 581
7.121.2.12 ramebuffer Stride()	. 581
7.121.2.13getBlueColor()	. 581
7.121.2.14getBlueFromColor()	. 581
7.121.2.15getColorFrom24BitRGB()	. 582
7.121.2.16getColorFromRGB()	. 582
7.121.2.17getFramebufferStride()	. 582
7.121.2.1&getGreenColor()	. 583
7.121.2.19getGreenFromColor()	. 583
7.121.2.20getRedColor()	. 583
7.121.2.21getRedFromColor()	. 584
7.121.2.22init()	. 584
7.121.2.23nextLine()	. 584
7.121.2.24nextPixel()	. 584
7.122LED Class Reference	. 585
7.122.1 Member Function Documentation	. 585
7.122.1.1 get()	. 585
7.122.1.2 init()	. 586

CONTENTS xIvii

7.122.1.3 off()	586
7.122.1.4 on()	586
7.122.1.5 toggle()	586
7.123Line Class Reference	586
7.123.1 Detailed Description	588
7.123.2 Member Enumeration Documentation	588
7.123.2.1 LINE_ENDING_STYLE	588
7.123.3 Constructor & Destructor Documentation	588
7.123.3.1 Line()	588
7.123.4 Member Function Documentation	589
7.123.4.1 drawCanvasWidget()	589
7.123.4.2 getEnd()	589
7.123.4.3 getLineEndingStyle()	589
7.123.4.4 getLineWidth() [1/2] 5	590
7.123.4.5 getLineWidth() [2/2]	590
7.123.4.6 getMinimalRect()	590
7.123.4.7 getStart()	591
7.123.4.8 setCapPrecision()	591
7.123.4.9 setEnd() [1/2]	592
7.123.4.10setEnd() [2/2]	592
7.123.4.11setLine()	592
7.123.4.12setLineEndingStyle()	593
7.123.4.13setLineWidth() [1/2]	593
7.123.4.14setLineWidth() [2/2]	594
7.123.4.15setStart() [1/2]	594
7.123.4.16setStart() [2/2]	595
7.123.4.17updateEnd() [1/2]	595
7.123.4.18updateEnd() [2/2]	596
7.123.4.19updateLengthAndAngle()	596
7.123.4.20updateLineWidth() [1/2]	596
7.123.4.21updateLineWidth() [2/2]	597
7.123.4.22updateStart() [1/2]	597
7.123.4.23updateStart() [2/2] 5	598
7.124LineProgress Class Reference	598
7.124.1 Detailed Description	599
7.124.2 Constructor & Destructor Documentation	599
7.124.2.1 LineProgress()	599
7.124.2.2 ~LineProgress()	300
7.124.3 Member Function Documentation	300
7.124.3.1 getAlpha()	300

xlviii CONTENTS

7.124.3.2 getEnd()	600
7.124.3.3 getLineEndingStyle()	600
7.124.3.4 getLineWidth()	600
7.124.3.5 getStart()	601
7.124.3.6 setAlpha()	601
7.124.3.7 setEnd()	601
7.124.3.8 setLineEndingStyle()	602
7.124.3.9 setLineWidth()	602
7.124.3.10setPainter()	602
7.124.3.11setProgressIndicatorPosition()	603
7.124.3.12setStart()	603
7.124.3.13setValue()	603
7.125ListLayout Class Reference	604
7.125.1 Detailed Description	604
7.125.2 Constructor & Destructor Documentation	604
7.125.2.1 ListLayout()	604
7.125.2.2 ~ListLayout()	605
7.125.3 Member Function Documentation	605
7.125.3.1 add()	605
7.125.3.2 getDirection()	605
7.125.3.3 getType()	605
7.125.3.4 insert()	606
7.125.3.5 remove()	606
7.125.3.6 removeAll()	606
7.125.3.7 setDirection()	606
7.126LockFreeDMA_Queue Class Reference	607
7.126.1 Detailed Description	607
7.126.2 Constructor & Destructor Documentation	607
7.126.2.1 LockFreeDMA_Queue()	608
7.126.3 Member Function Documentation	608
7.126.3.1 first()	608
7.126.3.2 isEmpty()	608
7.126.3.3 isFull()	608
7.126.3.4 pop()	609
7.126.3.5 pushCopyOf()	609
$7.127 Many Block Allocator < block_size, blocks, by tes_pr_pixel > Class\ Template\ Reference \ .\ .\ .\ .\ .$	609
7.127.1 Detailed Description	609
7.127.2 Member Function Documentation	610
7.127.2.1 allocateBlock()	610
7.127.2.2 freeBlockAfterTransfer()	610

CONTENTS xlix

7.127.2.3 getBlockForTransfer()	610
7.127.2.4 hasBlockReadyForTransfer()	611
7.127.2.5 markBlockReadyForTransfer()	611
7.128Matrix4x4 Class Reference	611
7.128.1 Detailed Description	612
7.128.2 Constructor & Destructor Documentation	612
7.128.2.1 Matrix4x4()	612
7.128.3 Member Function Documentation	612
7.128.3.1 concatenateXRotation()	612
7.128.3.2 concatenateXScale()	613
7.128.3.3 concatenateXTranslation()	613
7.128.3.4 concatenateYRotation()	613
7.128.3.5 concatenateYScale()	613
7.128.3.6 concatenateYTranslation()	614
7.128.3.7 concatenateZRotation()	614
7.128.3.8 concatenateZScale()	614
7.128.3.9 concatenateZTranslation()	615
7.128.3.10getElement()	615
7.128.3.11identity()	615
7.128.3.12setElement()	615
7.128.3.13setViewDistance()	616
7.129MCUInstrumentation Class Reference	616
7.129.1 Detailed Description	617
7.129.2 Constructor & Destructor Documentation	617
7.129.2.1 MCUInstrumentation()	617
7.129.2.2 ~MCUInstrumentation()	617
7.129.3 Member Function Documentation	617
7.129.3.1 getCCConsumed()	617
7.129.3.2 getCPUCycles()	617
7.129.3.3 getElapsedUS()	618
7.129.3.4 init()	618
7.129.3.5 setCCConsumed()	618
7.129.3.6 setMCUActive()	618
7.130ModalWindow Class Reference	618
7.130.1 Detailed Description	620
7.130.2 Constructor & Destructor Documentation	620
7.130.2.1 ModalWindow()	620
7.130.2.2 ~ModalWindow()	620
7.130.3 Member Function Documentation	620
7.130.3.1 add()	620

I CONTENTS

7.130.3.2 getBackgroundHeight()	. 620
7.130.3.3 getBackgroundWidth()	. 621
7.130.3.4 getShadeAlpha()	. 621
7.130.3.5 getShadeColor()	. 621
7.130.3.6 hide()	. 621
7.130.3.7 isShowing()	. 621
7.130.3.8 remove()	. 622
7.130.3.9 setBackground() [1/2]	. 622
7.130.3.10setBackground() [2/2]	. 622
7.130.3.11setShadeAlpha()	. 622
7.130.3.12etShadeColor()	. 623
7.130.3.13show()	. 623
$7.131 Move Animator < T > Class Template Reference \dots $. 623
7.131.1 Detailed Description	. 624
7.131.2 Constructor & Destructor Documentation	. 624
7.131.2.1 MoveAnimator()	. 625
7.131.2.2 ~MoveAnimator()	. 625
7.131.3 Member Function Documentation	. 625
7.131.3.1 cancelMoveAnimation()	. 625
7.131.3.2 clearMoveAnimationEndedAction()	. 625
7.131.3.3 getMoveAnimationDelay()	. 625
7.131.3.4 handleTickEvent()	. 625
7.131.3.5 isMoveAnimationRunning()	. 625
7.131.3.6 isRunning()	. 626
7.131.3.7 nextMoveAnimationStep()	. 626
7.131.3.8 setMoveAnimationDelay()	. 626
7.131.3.9 setMoveAnimationEndedAction()	. 626
7.131.3.10startMoveAnimation()	. 626
7.132MVPApplication Class Reference	. 627
7.132.1 Detailed Description	. 627
7.132.2 Constructor & Destructor Documentation	. 628
7.132.2.1 MVPApplication()	. 628
7.132.2.2 ~MVPApplication()	. 628
7.132.3 Member Function Documentation	. 628
7.132.3.1 evaluatePendingScreenTransition()	. 628
7.132.3.2 handlePendingScreenTransition()	. 628
7.133MVPHeap Class Reference	. 628
7.133.1 Detailed Description	. 629
7.133.2 Constructor & Destructor Documentation	. 629
7.133.2.1 MVPHeap()	. 629

7.133.2.2 ~MVPHeap()	629
7.134NoDMA Class Reference	629
7.134.1 Detailed Description	630
7.134.2 Constructor & Destructor Documentation	630
7.134.2.1 NoDMA()	630
7.134.3 Member Function Documentation	630
7.134.3.1 flush()	630
7.134.3.2 getBlitCaps()	631
7.134.3.3 setupDataCopy()	631
7.134.3.4 setupDataFill()	631
7.134.3.5 signalDMAInterrupt()	631
7.135NoTouchController Class Reference	631
7.135.1 Detailed Description	632
7.135.2 Constructor & Destructor Documentation	632
7.135.2.1 ~NoTouchController()	632
7.135.3 Member Function Documentation	632
7.135.3.1 init()	632
7.135.3.2 sampleTouch()	632
7.136NoTransition Class Reference	633
7.136.1 Detailed Description	633
7.136.2 Constructor & Destructor Documentation	633
7.136.2.1 NoTransition()	633
7.136.2.2 ~NoTransition()	633
7.136.3 Member Function Documentation	633
7.136.3.1 handleTickEvent()	633
7.137OSWrappers Class Reference	634
7.137.1 Detailed Description	634
7.137.2 Member Function Documentation	634
7.137.2.1 giveFrameBufferSemaphore()	634
7.137.2.2 giveFrameBufferSemaphoreFromISR()	634
7.137.2.3 initialize()	634
7.137.2.4 signalVSync()	635
7.137.2.5 takeFrameBufferSemaphore()	635
7.137.2.6 taskDelay()	635
7.137.2.7 tryTakeFrameBufferSemaphore()	635
7.137.2.8 waitForVSync()	635
7.138Outline Class Reference	636
7.138.1 Detailed Description	636
7.138.2 Member Typedef Documentation	637
7.138.2.1 OutlineFlags_t	637

lii CONTENTS

7.138.3 Constructor & Destructor Documentation	. 637
7.138.3.1 Outline()	. 637
7.138.3.2 ~Outline()	. 637
7.138.4 Member Function Documentation	. 637
7.138.4.1 getCells()	. 637
7.138.4.2 getNumCells()	. 637
7.138.4.3 lineTo()	. 638
7.138.4.4 moveTo()	. 638
7.138.4.5 reset()	. 638
7.138.4.6 setMaxRenderY()	. 638
7.138.4.7 wasOutlineTooComplex()	. 638
7.139 Painter ABGR 2222 Class Reference	. 639
7.139.1 Detailed Description	. 639
7.139.2 Constructor & Destructor Documentation	. 640
7.139.2.1 PainterABGR2222()	. 640
7.139.3 Member Function Documentation	. 640
7.139.3.1 getAlpha()	. 640
7.139.3.2 getColor()	. 640
7.139.3.3 render()	. 640
7.139.3.4 renderNext()	. 641
7.139.3.5 setAlpha()	. 641
7.139.3.6 setColor()	. 642
7.140 Painter ABGR 2222 Bitmap Class Reference	. 642
7.140.1 Detailed Description	. 643
7.140.2 Constructor & Destructor Documentation	. 643
7.140.2.1 PainterABGR2222Bitmap()	. 643
7.140.3 Member Function Documentation	. 643
7.140.3.1 getAlpha()	. 643
7.140.3.2 render()	. 643
7.140.3.3 renderInit()	. 644
7.140.3.4 renderNext()	. 644
7.140.3.5 setAlpha()	. 645
7.140.3.6 setBitmap()	. 645
7.141 Painter ARGB2222 Class Reference	. 645
7.141.1 Detailed Description	. 646
7.141.2 Constructor & Destructor Documentation	. 646
7.141.2.1 PainterARGB2222()	. 646
7.141.3 Member Function Documentation	. 646
7.141.3.1 getAlpha()	. 646
7.141.3.2 getColor()	. 647

7.141.3.3 render()
7.141.3.4 renderNext()
7.141.3.5 setAlpha()
7.141.3.6 setColor()
7.142PainterARGB2222Bitmap Class Reference
7.142.1 Detailed Description
7.142.2 Constructor & Destructor Documentation
7.142.2.1 PainterARGB2222Bitmap()
7.142.3 Member Function Documentation
7.142.3.1 getAlpha()
7.142.3.2 render()
7.142.3.3 renderInit()
7.142.3.4 renderNext()
7.142.3.5 setAlpha()
7.142.3.6 setBitmap()
7.143PainterARGB8888 Class Reference
7.143.1 Detailed Description
7.143.2 Constructor & Destructor Documentation
7.143.2.1 PainterARGB8888()
7.143.3 Member Function Documentation
7.143.3.1 getAlpha()
7.143.3.2 getColor()
7.143.3.3 render()
7.143.3.4 renderNext()
7.143.3.5 setAlpha()
7.143.3.6 setColor()
7.144PainterARGB8888Bitmap Class Reference
7.144.1 Detailed Description
7.144.2 Constructor & Destructor Documentation
7.144.2.1 PainterARGB8888Bitmap()
7.144.3 Member Function Documentation
7.144.3.1 getAlpha()
7.144.3.2 render()
7.144.3.3 renderInit()
7.144.3.4 renderNext()
7.144.3.5 setAlpha()
7.144.3.6 setBitmap()
7.145PainterARGB8888L8Bitmap Class Reference
7.145.1 Detailed Description
7.145.2 Constructor & Destructor Documentation

liv CONTENTS

7.145.2.1 PainterARGB8888L8Bitmap()
7.145.3 Member Function Documentation
7.145.3.1 getAlpha()
7.145.3.2 render()
7.145.3.3 renderInit()
7.145.3.4 renderNext()
7.145.3.5 setAlpha()
7.145.3.6 setBitmap()
7.146PainterBGRA2222 Class Reference
7.146.1 Detailed Description
7.146.2 Constructor & Destructor Documentation
7.146.2.1 PainterBGRA2222()
7.146.3 Member Function Documentation
7.146.3.1 getAlpha()
7.146.3.2 getColor()
7.146.3.3 render()
7.146.3.4 renderNext()
7.146.3.5 setAlpha()
7.146.3.6 setColor()
7.147PainterBGRA2222Bitmap Class Reference
7.147.1 Detailed Description
7.147.2 Constructor & Destructor Documentation
7.147.2.1 PainterBGRA2222Bitmap()
7.147.3 Member Function Documentation
7.147.3.1 getAlpha()
7.147.3.2 render()
7.147.3.3 renderInit()
7.147.3.4 renderNext()
7.147.3.5 setAlpha()
7.147.3.6 setBitmap()
7.148PainterBW Class Reference
7.148.1 Detailed Description
7.148.2 Member Function Documentation
7.148.2.1 bw()
7.148.2.2 getColor()
7.148.2.3 render()
7.148.2.4 renderNext()
7.148.2.5 setColor()
7.149PainterBWBitmap Class Reference
7.149.1 Detailed Description

7.149.2 Constructor & Destructor Documentation	′1
7.149.2.1 PainterBWBitmap()	'1
7.149.3 Member Function Documentation	'2
7.149.3.1 render()	'2
7.149.3.2 renderInit()	'2
7.149.3.3 renderNext()	'2
7.149.3.4 setBitmap()	'3
7.150 Painter GRAY2 Class Reference	'3
7.150.1 Detailed Description	'4
7.150.2 Constructor & Destructor Documentation	'4
7.150.2.1 PainterGRAY2()	'4
7.150.3 Member Function Documentation	'4
7.150.3.1 getAlpha()	'4
7.150.3.2 getColor()	'4
7.150.3.3 render()	'5
7.150.3.4 renderNext()	'5
7.150.3.5 setAlpha()	'5
7.150.3.6 setColor()	'6
7.151 Painter GRAY2 Bitmap Class Reference	'6
7.151.1 Detailed Description	7
7.151.2 Constructor & Destructor Documentation	7
7.151.2.1 PainterGRAY2Bitmap()	7
7.151.3 Member Function Documentation	7
7.151.3.1 getAlpha()	7
7.151.3.2 render()	'8
7.151.3.3 renderInit()	'8
7.151.3.4 renderNext()	'8
7.151.3.5 setAlpha()	'9
7.151.3.6 setBitmap()	'9
7.152PainterGRAY4 Class Reference	'9
7.152.1 Detailed Description	30
7.152.2 Constructor & Destructor Documentation	30
7.152.2.1 PainterGRAY4()	30
7.152.3 Member Function Documentation	30
7.152.3.1 getAlpha()	31
7.152.3.2 getColor()	31
7.152.3.3 render()	31
7.152.3.4 renderNext()	2
7.152.3.5 setAlpha()	2
7.152.3.6 setColor()	32

Ivi CONTENTS

7.153 Painter GRAY4 Bitmap Class Reference	382
7.153.1 Detailed Description	383
7.153.2 Constructor & Destructor Documentation	383
7.153.2.1 PainterGRAY4Bitmap()	384
7.153.3 Member Function Documentation	384
7.153.3.1 getAlpha()	384
7.153.3.2 render()	384
7.153.3.3 renderInit()	385
7.153.3.4 renderNext()	385
7.153.3.5 setAlpha()	385
7.153.3.6 setBitmap()	385
7.154PainterRGB565 Class Reference	386
7.154.1 Detailed Description	387
7.154.2 Constructor & Destructor Documentation	387
7.154.2.1 PainterRGB565()	387
7.154.3 Member Function Documentation	387
7.154.3.1 getAlpha()	387
7.154.3.2 getColor()	387
7.154.3.3 render()	388
7.154.3.4 renderNext()	388
7.154.3.5 setAlpha()	389
7.154.3.6 setColor()	389
7.155PainterRGB565Bitmap Class Reference	389
7.155.1 Detailed Description	390
7.155.2 Constructor & Destructor Documentation	390
7.155.2.1 PainterRGB565Bitmap()	390
7.155.3 Member Function Documentation	390
7.155.3.1 getAlpha()	390
7.155.3.2 render()	391
7.155.3.3 renderInit()	391
7.155.3.4 renderNext()	391
7.155.3.5 setAlpha()	392
7.155.3.6 setBitmap()	392
7.156PainterRGB565L8Bitmap Class Reference	392
7.156.1 Detailed Description	393
7.156.2 Constructor & Destructor Documentation	393
7.156.2.1 PainterRGB565L8Bitmap()	393
7.156.3 Member Function Documentation	394
7.156.3.1 getAlpha()	394
7.156.3.2 render()	394

7.156.3.3 renderInit()	95
7.156.3.4 renderNext()	95
7.156.3.5 setAlpha()	95
7.156.3.6 setBitmap()	95
7.157PainterRGB888 Class Reference	96
7.157.1 Detailed Description	96
7.157.2 Constructor & Destructor Documentation	97
7.157.2.1 PainterRGB888()	97
7.157.3 Member Function Documentation	97
7.157.3.1 getAlpha()	97
7.157.3.2 getColor()	97
7.157.3.3 render()	98
7.157.3.4 renderNext()	98
7.157.3.5 setAlpha()	98
7.157.3.6 setColor()	00
7.158PainterRGB888Bitmap Class Reference	00
7.158.1 Detailed Description	01
7.158.2 Constructor & Destructor Documentation	01
7.158.2.1 PainterRGB888Bitmap()	01
7.158.3 Member Function Documentation	01
7.158.3.1 getAlpha()	01
7.158.3.2 render()	02
7.158.3.3 renderInit()	02
7.158.3.4 renderNext()	02
7.158.3.5 setAlpha()	03
7.158.3.6 setBitmap()	03
7.159PainterRGB888L8Bitmap Class Reference	03
7.159.1 Detailed Description	04
7.159.2 Constructor & Destructor Documentation	04
7.159.2.1 PainterRGB888L8Bitmap()	04
7.159.3 Member Function Documentation	05
7.159.3.1 getAlpha()	05
7.159.3.2 render()	05
7.159.3.3 renderInit()	05
7.159.3.4 renderNext()	06
7.159.3.5 setAlpha()	06
7.159.3.6 setBitmap()	06
7.160 Painter RGBA2222 Class Reference	07
7.160.1 Detailed Description	07
7.160.2 Constructor & Destructor Documentation	80

Iviii CONTENTS

7.160.2.1 PainterRGBA2222()
7.160.3 Member Function Documentation
7.160.3.1 getAlpha()
7.160.3.2 getColor()
7.160.3.3 render()
7.160.3.4 renderNext()
7.160.3.5 setAlpha()
7.160.3.6 setColor()
7.161 Painter RGBA2222 Bitmap Class Reference
7.161.1 Detailed Description
7.161.2 Constructor & Destructor Documentation
7.161.2.1 PainterRGBA2222Bitmap()
7.161.3 Member Function Documentation
7.161.3.1 getAlpha()
7.161.3.2 render()
7.161.3.3 renderInit()
7.161.3.4 renderNext()
7.161.3.5 setAlpha()
7.161.3.6 setBitmap()
7.162Pair < T1, T2 > Struct Template Reference
7.162.1 Detailed Description
7.162.2 Constructor & Destructor Documentation
7.162.2.1 Pair() [1/3]
7.162.2.2 Pair() [2/3]
7.162.2.3 Pair() [3/3]
7.163Partition < ListOfTypes, NUMBER_OF_ELEMENTS > Class Template Reference
7.163.1 Detailed Description
7.163.2 Member Typedef Documentation
7.163.2.1 SupportedTypesList
7.163.3 Member Enumeration Documentation
7.163.3.1 anonymous enum
7.163.4 Constructor & Destructor Documentation
7.163.4.1 Partition()
7.163.5 Member Function Documentation
7.163.5.1 capacity()
7.163.5.2 element() [1/2]
7.163.5.3 element() [2/2]
7.163.5.4 element_size()
7.164PixelDataWidget Class Reference
7.164.1 Detailed Description

7.164.2 Constructor & Destructor Documentation	18
7.164.2.1 PixelDataWidget()	18
7.164.3 Member Function Documentation	19
7.164.3.1 draw()	19
7.164.3.2 getAlpha()	19
7.164.3.3 getSolidRect()	19
7.164.3.4 setAlpha()	19
7.164.3.5 setBitmapFormat()	20
7.164.3.6 setPixelData()	20
7.165 Point Struct Reference	20
7.165.1 Member Function Documentation	20
7.165.1.1 dist_sqr()	21
7.166 Point3D Struct Reference	21
7.167 Point4 Class Reference	21
7.167.1 Detailed Description	21
7.167.2 Constructor & Destructor Documentation	22
7.167.2.1 Point4() [1/2]	22
7.167.2.2 Point4() [2/2]	22
7.168PreRenderable < T > Class Template Reference	22
7.168.1 Detailed Description	22
7.168.2 Constructor & Destructor Documentation	23
7.168.2.1 PreRenderable()	23
7.168.3 Member Function Documentation	23
7.168.3.1 draw()	23
7.168.3.2 isPreRendered()	23
7.168.3.3 preRender()	24
7.168.3.4 setupDrawChain()	24
7.169Presenter Class Reference	24
7.169.1 Detailed Description	24
7.169.2 Constructor & Destructor Documentation	24
7.169.2.1 ~Presenter()	25
7.169.2.2 Presenter()	25
7.169.3 Member Function Documentation	25
7.169.3.1 activate()	25
7.169.3.2 deactivate()	25
7.170CWRUtil::Q10 Class Reference	25
7.170.1 Detailed Description	26
7.170.2 Constructor & Destructor Documentation	26
7.170.2.1 Q10() [1/2]	26
7.170.2.2 Q10() [2/2]	26

IX CONTENTS

7.170.3 Member Function Documentation	726
7.170.3.1 operator int()	726
7.170.3.2 operator*()	727
7.170.3.3 operator+()	727
7.170.3.4 operator-()	727
7.170.3.5 operator/()	727
7.170.3.6 to()	728
7.171 CWRUtil::Q15 Class Reference	728
7.171.1 Detailed Description	728
7.171.2 Constructor & Destructor Documentation	729
7.171.2.1 Q15()	729
7.171.3 Member Function Documentation	729
7.171.3.1 operator int()	729
7.171.3.2 operator+()	729
7.171.3.3 operator-()	729
7.171.3.4 operator/()	730
7.172CWRUtil::Q5 Class Reference	730
7.172.1 Detailed Description	731
7.172.2 Constructor & Destructor Documentation	731
7.172.2.1 Q5() [1/2]	731
7.172.2.2 Q5() [2/2]	731
7.172.3 Member Function Documentation	731
7.172.3.1 operator int()	732
7.172.3.2 operator*() [1/3]	732
7.172.3.3 operator*() [2/3]	732
7.172.3.4 operator*() [3/3]	733
7.172.3.5 operator+()	733
7.172.3.6 operator-() [1/2]	733
7.172.3.7 operator-() [2/2]	733
7.172.3.8 operator/() [1/2]	734
7.172.3.9 operator/() [2/2]	734
7.172.3.10to()	734
7.173Quadruple Class Reference	735
7.173.1 Detailed Description	735
7.173.2 Constructor & Destructor Documentation	736
7.173.2.1 Quadruple() [1/2]	736
7.173.2.2 Quadruple() [2/2]	736
7.173.3 Member Function Documentation	736
7.173.3.1 getElement()	736
7.173.3.2 getW()	736

7.173.3.3 getX()	'37
7.173.3.4 getY()	'37
7.173.3.5 getZ()	'37
7.173.3.6 setElement()	'37
7.173.3.7 setW()	'38
7.173.3.8 setX()	'38
7.173.3.9 setY()	'38
7.173.3.10setZ()	'38
7.174RadioButton Class Reference	'38
7.174.1 Detailed Description	' 40
7.174.2 Constructor & Destructor Documentation	' 40
7.174.2.1 RadioButton()	' 40
7.174.2.2 ~RadioButton()	' 40
7.174.3 Member Function Documentation	' 40
7.174.3.1 draw()	' 40
7.174.3.2 getAlpha()	' 40
7.174.3.3 getCurrentlyDisplayedBitmap()	' 41
7.174.3.4 getDeselectionEnabled()	' 41
7.174.3.5 getSelected()	' 41
7.174.3.6 getSolidRect()	' 41
7.174.3.7 getType()	' 42
7.174.3.8 handleClickEvent()	' 42
7.174.3.9 setAlpha()	' 42
7.174.3.10setBitmaps()	' 42
7.174.3.11setDeselectedAction()	' 43
7.174.3.12setDeselectionEnabled()	743
7.174.3.13setSelected()	' 43
7.175 Radio Button Group < CAPACITY > Class Template Reference	' 43
7.175.1 Detailed Description	' 45
7.175.2 Constructor & Destructor Documentation	'45
7.175.2.1 RadioButtonGroup()	' 45
7.175.2.2 ~RadioButtonGroup()	' 45
7.175.3 Member Function Documentation	'45
7.175.3.1 add()	' 45
7.175.3.2 getDeselectionEnabled()	' 46
7.175.3.3 getRadioButton()	' 46
7.175.3.4 getSelectedRadioButton()	' 46
7.175.3.5 getSelectedRadioButtonIndex()	' 46
7.175.3.6 radioButtonClickedHandler()	' 46
7.175.3.7 radioButtonDeselectedHandler()	' 47

lxii CONTENTS

7.175.3.8 setDeselectionEnabled()	747
7.175.3.9 setRadioButtonDeselectedHandler()	747
7.175.3.10setRadioButtonSelectedHandler()	747
7.175.3.11setSelected()	749
7.176Rasterizer Class Reference	749
7.176.1 Detailed Description	750
7.176.2 Member Enumeration Documentation	750
7.176.2.1 anonymous enum	750
7.176.2.2 anonymous enum	750
7.176.2.3 FillingRule	751
7.176.3 Constructor & Destructor Documentation	751
7.176.3.1 Rasterizer()	751
7.176.4 Member Function Documentation	751
7.176.4.1 calculateAlpha()	751
7.176.4.2 lineTo()	752
7.176.4.3 moveTo()	752
7.176.4.4 render()	752
7.176.4.5 reset()	752
7.176.4.6 setFillingRule()	753
7.176.4.7 setMaxRenderY()	753
7.176.4.8 wasOutlineTooComplex()	753
7.177Rect Class Reference	753
7.177.1 Detailed Description	754
7.177.2 Constructor & Destructor Documentation	754
7.177.2.1 Rect() [1/2]	754
7.177.2.2 Rect() [2/2]	755
7.177.3 Member Function Documentation	755
7.177.3.1 area()	755
7.177.3.2 bottom()	755
7.177.3.3 expandToFit()	755
7.177.3.4 includes()	756
7.177.3.5 intersect() [1/2]	756
7.177.3.6 intersect() [2/2]	756
7.177.3.7 isEmpty()	757
7.177.3.8 operator"!=()	757
7.177.3.9 operator&()	757
7.177.3.10operator&=()	757
7.177.3.11operator==()	758
7.177.3.12right()	758
7.178Renderer Class Reference	758

7.178.1 Detailed Description	⁷ 58
7.178.2 Constructor & Destructor Documentation	⁷ 59
7.178.2.1 Renderer() [1/2]	⁷ 59
7.178.2.2 Renderer() [2/2] 7	⁷ 59
7.178.3 Member Function Documentation	⁷ 59
7.178.3.1 getRenderingBuffer()	⁷ 59
7.178.3.2 render()	⁷ 59
7.178.3.3 setRenderingBuffer()	⁷ 60
7.179RenderingBuffer Class Reference	⁷ 60
7.179.1 Detailed Description	⁷ 60
7.179.2 Constructor & Destructor Documentation	⁷ 61
7.179.2.1 RenderingBuffer() [1/2]	⁷ 61
7.179.2.2 ~RenderingBuffer()	⁷ 61
7.179.2.3 RenderingBuffer() [2/2]	⁷ 61
7.179.3 Member Function Documentation	⁷ 62
7.179.3.1 attach()	⁷ 62
7.179.3.2 getHeight()	⁷ 62
7.179.3.3 getWidth()	⁷ 62
7.179.3.4 getXAdjust()	⁷ 62
7.179.3.5 inbox()	⁷ 63
7.179.3.6 row() [1/2]	⁷ 63
7.179.3.7 row() [2/2]	⁷ 63
7.180 RepeatButton Class Reference	⁷ 64
7.180.1 Detailed Description	⁷ 64
7.180.2 Constructor & Destructor Documentation	⁷ 64
7.180.2.1 RepeatButton()	⁷ 64
7.180.3 Member Function Documentation	⁷ 64
7.180.3.1 getDelay()	⁷ 65
7.180.3.2 getInterval()	765
7.180.3.3 handleClickEvent()	⁷ 65
7.180.3.4 handleTickEvent()	⁷ 65
7.180.3.5 setDelay()	⁷ 65
7.180.3.6 setInterval()	⁷ 66
7.181RepeatButtonTrigger Class Reference	⁷ 66
7.181.1 Detailed Description	767
7.181.2 Member Function Documentation	⁷ 67
7.181.2.1 getDelay()	⁷ 67
7.181.2.2 getInterval()	767
7.181.2.3 handleClickEvent()	
7.181.2.4 setDelay()	767

lxiv CONTENTS

7.181.2.5 setInterval()	'68
7.182ScalableImage Class Reference	'68
7.182.1 Detailed Description	'69
7.182.2 Member Enumeration Documentation	' 69
7.182.2.1 ScalingAlgorithm	'69
7.182.3 Constructor & Destructor Documentation	' 69
7.182.3.1 ScalableImage()	' 69
7.182.3.2 ~ScalableImage()	70
7.182.4 Member Function Documentation	'70
7.182.4.1 draw()	'70
7.182.4.2 drawTriangle()	'70
7.182.4.3 getAlpha()	' 71
7.182.4.4 getBitmap()	71
7.182.4.5 getScalingAlgorithm()	' 71
7.182.4.6 getSolidRect()	⁷ 71
7.182.4.7 getType()	' 71
7.182.4.8 lookupRenderVariant()	'72
7.182.4.9 setAlpha()	'72
7.182.4.10setBitmap()	'72
7.182.4.11setScalingAlgorithm()	'72
7.183Scanline Class Reference	′73
7.183.1 Detailed Description	′73
7.183.2 Constructor & Destructor Documentation	'74
7.183.2.1 Scanline()	' 74
7.183.2.2 ~Scanline()	' 74
7.183.3 Member Function Documentation	' 74
7.183.3.1 addCell()	'74
7.183.3.2 addSpan()	′76
7.183.3.3 getNumSpans()	′76
7.183.3.4 getY()	′76
7.183.3.5 isReady()	′76
7.183.3.6 reset()	' 77
7.183.3.7 resetSpans()	' 77
7.184Screen Class Reference	' 77
7.184.1 Detailed Description	′78
7.184.2 Constructor & Destructor Documentation	′78
7.184.2.1 Screen()	′78
7.184.2.2 ~Screen()	'79
7.184.3 Member Function Documentation	'79
7.184.3.1 add()	′79

7.184.3.2 afterTransition()	 779
7.184.3.3 bindTransition()	 779
7.184.3.4 draw() [1/2]	 779
7.184.3.5 draw() [2/2]	 780
7.184.3.6 getRootContainer()	 780
7.184.3.7 handleClickEvent()	 780
7.184.3.8 handleDragEvent()	 780
7.184.3.9 handleGestureEvent()	 781
7.184.3.10handleKeyEvent()	 781
7.184.3.11handleTickEvent()	 781
7.184.3.12JSMOC()	 781
7.184.3.13remove()	 782
7.184.3.14setupScreen()	 782
7.184.3.15startSMOC()	 782
7.184.3.1@earDownScreen()	 782
7.184.3.17useSMOCDrawing()	 782
7.184.3.1&usingSMOC()	 783
7.185ScrollableContainer Class Reference	 783
7.185.1 Detailed Description	 786
7.185.2 Constructor & Destructor Documentation	 786
7.185.2.1 ScrollableContainer()	 786
7.185.2.2 ~ScrollableContainer()	 786
7.185.3 Member Function Documentation	 786
7.185.3.1 add()	 786
7.185.3.2 childGeometryChanged()	 787
7.185.3.3 doScroll()	 787
7.185.3.4 enableHorizontalScroll()	 787
7.185.3.5 enableVerticalScroll()	 788
7.185.3.6 getContainedArea()	 788
7.185.3.7 getLastChild()	 788
7.185.3.8 getScrolledX()	 788
7.185.3.9 getScrolledY()	 789
7.185.3.10getType()	 789
7.185.3.11getXBorder()	 789
7.185.3.12getXScrollbar()	 789
7.185.3.13getYBorder()	 790
7.185.3.14getYScrollbar()	 790
7.185.3.15handleClickEvent()	 790
7.185.3.16handleDragEvent()	 790
7.185.3.17handleGestureEvent()	 791

lxvi CONTENTS

7.185.3.18handleTickEvent()	791
7.185.3.19nvalidateScrollbars()	791
7.185.3.20sScrollableXY()	791
7.185.3.21moveChildrenRelative()	792
7.185.3.22reset()	792
7.185.3.23setMaxVelocity()	792
7.185.3.24setScrollbarPadding()	792
7.185.3.25setScrollbarsAlpha()	792
7.185.3.26setScrollbarsColor()	793
7.185.3.27setScrollbarsPermanentlyVisible()	793
7.185.3.2&setScrollbarsVisible()	793
7.185.3.2%etScrollbarWidth()	793
7.185.3.30setScrollThreshold()	793
7.186ScrollBase Class Reference	794
7.186.1 Detailed Description	797
7.186.2 Member Enumeration Documentation	797
7.186.2.1 AnimationState	797
7.186.3 Constructor & Destructor Documentation	797
7.186.3.1 ScrollBase()	797
7.186.3.2 ~ScrollBase()	798
7.186.4 Member Function Documentation	798
7.186.4.1 allowHorizontalDrag()	798
7.186.4.2 allowVerticalDrag()	798
7.186.4.3 animateToItem()	798
7.186.4.4 animateToPosition()	799
7.186.4.5 getAnimationSteps()	799
7.186.4.6 getCircular()	799
7.186.4.7 getDragAcceleration()	800
7.186.4.8 getDrawableMargin()	800
7.186.4.9 getDrawableSize()	800
7.186.4.10getHorizontal()	800
7.186.4.11getMaxSwipeItems()	801
7.186.4.12getNearestAlignedOffset()	801
7.186.4.13getNormalizedOffset()	801
7.186.4.14getNumberOfItems()	801
7.186.4.15getOffset()	802
7.186.4.16getPositionForItem()	802
7.186.4.17getSwipeAcceleration()	802
7.186.4.18handleDragEvent()	803
7.186.4.19handleGestureEvent()	803

Ixviii CONTENTS

7.188.1 Detailed Description	815
7.188.2 Constructor & Destructor Documentation	815
7.188.2.1 ScrollWheel()	815
7.188.2.2 ~ScrollWheel()	815
7.188.3 Member Function Documentation	815
7.188.3.1 setDrawables()	815
7.189ScrollWheelBase Class Reference	816
7.189.1 Detailed Description	816
7.189.2 Constructor & Destructor Documentation	817
7.189.2.1 ScrollWheelBase()	817
7.189.2.2 ~ScrollWheelBase()	817
7.189.3 Member Function Documentation	817
7.189.3.1 animateToPosition()	817
7.189.3.2 getPositionForItem()	817
7.189.3.3 getSelectedItem()	818
7.189.3.4 getSelectedItemOffset()	818
7.189.3.5 handleClickEvent()	818
7.189.3.6 handleDragEvent()	818
7.189.3.7 handleGestureEvent()	819
7.189.3.8 keepOffsetInsideLimits()	819
7.189.3.9 setAnimateToCallback()	819
7.189.3.10setSelectedItemOffset()	819
7.190ScrollWheelWithSelectionStyle Class Reference	820
7.190.1 Detailed Description	821
7.190.2 Constructor & Destructor Documentation	821
7.190.2.1 ScrollWheelWithSelectionStyle()	822
$7.190.2.2 \sim ScrollWheelWithSelectionStyle() \qquad . \qquad $	822
7.190.3 Member Function Documentation	822
7.190.3.1 getSelectedItemExtraSizeAfter()	822
7.190.3.2 getSelectedItemExtraSizeBefore()	822
7.190.3.3 getSelectedItemMarginAfter()	822
7.190.3.4 getSelectedItemMarginBefore()	823
7.190.3.5 initialize()	823
7.190.3.6 itemChanged()	823
7.190.3.7 refreshDrawableListsLayout()	823
7.190.3.8 setCircular()	824
7.190.3.9 setDrawables()	824
7.190.3.10setDrawableSize()	824
7.190.3.11setHeight()	825
7.190.3.12setHorizontal()	825

7.190.3.13setNumberOfItems()	825
7.190.3.14setOffset()	826
7.190.3.15setSelectedItemExtraSize()	826
7.190.3.16setSelectedItemMargin()	826
7.190.3.17setSelectedItemOffset()	827
7.190.3.1&setSelectedItemPosition()	827
7.190.3.19setWidth()	828
7.191SDL2TouchController Class Reference	828
7.191.1 Detailed Description	828
7.191.2 Member Function Documentation	828
7.191.2.1 init()	828
7.191.2.2 sampleTouch()	828
7.192SDLTouchController Class Reference	829
7.192.1 Detailed Description	829
7.192.2 Member Function Documentation	829
7.192.2.1 init()	829
7.192.2.2 sampleTouch()	829
7.193Shape< POINTS > Class Template Reference	830
7.193.1 Detailed Description	830
7.193.2 Constructor & Destructor Documentation	831
7.193.2.1 ~Shape()	831
7.193.3 Member Function Documentation	831
7.193.3.1 getCacheX()	831
7.193.3.2 getCacheY()	831
7.193.3.3 getCornerX()	832
7.193.3.4 getCornerY()	832
7.193.3.5 getNumPoints()	832
7.193.3.6 setCache()	833
7.193.3.7 setCorner()	833
7.194AbstractShape::ShapePoint< T > Struct Template Reference	833
7.194.1 Detailed Description	833
7.195SingleBlockAllocator< block_size, bytes_pr_pixel > Class Template Reference	834
7.195.1 Detailed Description	834
7.195.2 Member Function Documentation	834
7.195.2.1 allocateBlock()	835
7.195.2.2 freeBlockAfterTransfer()	835
7.195.2.3 getBlockForTransfer()	835
7.195.2.4 hasBlockReadyForTransfer()	836
7.195.2.5 markBlockReadyForTransfer()	836
7.196SlideMenu Class Reference	836

IXX CONTENTS

7.196.1 Detailed Description	. 839
7.196.2 Member Function Documentation	. 839
7.196.2.1 add()	. 839
7.196.2.2 animateToState()	. 839
7.196.2.3 animationEndedHandler()	. 839
7.196.2.4 getAnimationDuration()	. 840
7.196.2.5 getAnimationEasingEquation()	. 840
7.196.2.6 getBackgroundX()	. 840
7.196.2.7 getBackgroundY()	. 840
7.196.2.8 getCollapsedXCoordinate()	. 840
7.196.2.9 getCollapsedYCoordinate()	. 840
7.196.2.10getExpandDirection()	. 841
7.196.2.11getExpandedStateTimeout()	. 841
7.196.2.12getExpandedStateTimer()	. 841
7.196.2.13getExpandedXCoordinate()	. 841
7.196.2.14getExpandedYCoordinate()	. 841
7.196.2.15getHiddenPixelsWhenExpanded()	. 841
7.196.2.16getState()	. 842
7.196.2.17getStateChangeButtonX()	. 842
7.196.2.18getStateChangeButtonY()	. 842
7.196.2.19getVisiblePixelsWhenCollapsed()	
7.196.2.20remove()	
7.196.2.21resetExpandedStateTimer()	. 843
7.196.2.22setAnimationDuration()	. 843
7.196.2.23setAnimationEasingEquation()	. 843
7.196.2.24setExpandDirection()	. 843
7.196.2.25setExpandedStateTimeout()	
7.196.2.26setHiddenPixelsWhenExpanded()	
7.196.2.27setState()	. 844
7.196.2.2&etStateChangedAnimationEndedCallback()	
7.196.2.2%etStateChangedCallback()	
7.196.2.30setup() [1/2]	
7.196.2.31setup() [2/2]	. 845
7.196.2.32setVisiblePixelsWhenCollapsed()	
7.196.2.33stateChangeButtonClickedHandler()	
7.197Slider Class Reference	
7.197.1 Detailed Description	
7.197.2 Constructor & Destructor Documentation	
7.197.2.1 Slider()	
7.197.3 Member Function Documentation	. 848

7.197.3.1 getIndicatorMax()	. 848
7.197.3.2 getIndicatorMin()	. 849
7.197.3.3 getIndicatorPositionRangeSize()	. 849
7.197.3.4 getIndicatorRadius()	. 849
7.197.3.5 getMaxValue()	. 849
7.197.3.6 getMinValue()	. 850
7.197.3.7 getType()	. 850
7.197.3.8 getValue()	. 850
7.197.3.9 getValueRangeSize()	. 850
7.197.3.10handleClickEvent()	. 850
7.197.3.11handleDragEvent()	. 851
7.197.3.12positionToValue()	. 851
7.197.3.13setBitmaps() [1/2]	. 851
7.197.3.14setBitmaps() [2/2]	. 852
7.197.3.15setNewValueCallback()	. 852
7.197.3.16setStartValueCallback()	. 852
7.197.3.17setStopValueCallback()	. 853
7.197.3.18setupHorizontalSlider()	. 853
7.197.3.19setupVerticalSlider()	. 853
7.197.3.20setValue()	. 854
7.197.3.21setValueRange() [1/2]	. 854
7.197.3.2&etValueRange() [2/2]	. 855
7.197.3.23updateIndicatorPosition()	. 855
7.197.3.24valueToPosition()	. 855
7.198SlideTransition< templateDirection > Class Template Reference	. 856
7.198.1 Detailed Description	. 856
7.198.2 Constructor & Destructor Documentation	. 857
7.198.2.1 SlideTransition()	. 857
7.198.2.2 ~SlideTransition()	. 857
7.198.3 Member Function Documentation	. 857
7.198.3.1 handleTickEvent()	. 857
7.198.3.2 init()	. 857
7.198.3.3 initMoveDrawable()	. 858
7.198.3.4 tearDown()	. 858
7.198.3.5 tickMoveDrawable()	. 858
7.199Snapper< T > Class Template Reference	. 858
7.199.1 Detailed Description	. 859
7.199.2 Constructor & Destructor Documentation	. 859
7.199.2.1 Snapper()	. 859
7.199.2.2 ~Snapper()	. 859

Ixxii CONTENTS

7.199.3 Member Function Documentation	59
7.199.3.1 handleClickEvent()	59
7.199.3.2 handleDragEvent()	60
7.199.3.3 setDragAction()	60
7.199.3.4 setSnappedAction()	60
7.199.3.5 setSnapPosition()	61
7.200SnapshotWidget Class Reference	61
7.200.1 Detailed Description	62
7.200.2 Constructor & Destructor Documentation	62
7.200.2.1 SnapshotWidget()	62
7.200.2.2 ~SnapshotWidget()	62
7.200.3 Member Function Documentation	62
7.200.3.1 draw()	62
7.200.3.2 getAlpha()	62
7.200.3.3 getSolidRect()	63
7.200.3.4 getType()	63
7.200.3.5 makeSnapshot() [1/2]	63
7.200.3.6 makeSnapshot() [2/2]	63
7.200.3.7 setAlpha()	63
7.201LCD::StringVisuals Struct Reference	64
7.201.1 Detailed Description	64
7.201.2 Constructor & Destructor Documentation	64
7.201.2.1 StringVisuals() [1/2]	65
7.201.2.2 StringVisuals() [2/2]	65
7.202SwipeContainer Class Reference	65
7.202.1 Detailed Description	66
7.202.2 Member Function Documentation	66
7.202.2.1 add()	66
7.202.2.2 getNumberOfPages()	66
7.202.2.3 handleClickEvent()	67
7.202.2.4 handleDragEvent()	67
7.202.2.5 handleGestureEvent()	67
7.202.2.6 handleTickEvent()	67
7.202.2.7 remove()	68
7.202.2.8 setEndSwipeElasticWidth()	68
7.202.2.9 setPageIndicatorBitmaps()	68
7.202.2.10setPageIndicatorXY()	68
7.202.2.11setPageIndicatorXYWithCenteredX()	69
7.202.2.12setSelectedPage()	69
7.202.2.13setSwipeCutoff()	69

CONTENTS

7.203TextArea Class Reference	70
7.203.1 Detailed Description	71
7.203.2 Constructor & Destructor Documentation	71
7.203.2.1 TextArea()	71
7.203.3 Member Function Documentation	72
7.203.3.1 draw()	72
7.203.3.2 getAlpha()	72
7.203.3.3 getColor()	72
7.203.3.4 getIndentation()	72
7.203.3.5 getLinespacing()	73
7.203.3.6 getRotation()	73
7.203.3.7 getSolidRect()	73
7.203.3.8 getTextHeight()	73
7.203.3.9 getTextWidth()	73
7.203.3.10getType()	74
7.203.3.11getTypedText()	74
7.203.3.12getWideTextAction()	74
7.203.3.13resizeHeightToCurrentText()	74
7.203.3.14resizeToCurrentText()	75
7.203.3.15resizeToCurrentTextWithAlignment()	75
7.203.3.16setAlpha()	75
7.203.3.17setBaselineY()	76
7.203.3.18setColor()	76
7.203.3.19setIndentation()	76
7.203.3.20setLinespacing()	76
7.203.3.21setRotation()	77
7.203.3.22setTypedText()	77
7.203.3.23setWideTextAction()	77
7.203.3.24setXBaselineY()	78
7.204TextAreaWithOneWildcard Class Reference	78
7.204.1 Detailed Description	78
7.204.2 Constructor & Destructor Documentation	79
7.204.2.1 TextAreaWithOneWildcard()	79
7.204.3 Member Function Documentation	79
7.204.3.1 draw()	79
7.204.3.2 getTextHeight()	79
7.204.3.3 getTextWidth()	79
7.204.3.4 getType()	80
7.204.3.5 getWildcard()	80
7.204.3.6 setWildcard()	80

Ixxiv

7.205TextAreaWithTwoWildcards Class Reference	80
7.205.1 Detailed Description	81
7.205.2 Constructor & Destructor Documentation	81
7.205.2.1 TextAreaWithTwoWildcards()	81
7.205.3 Member Function Documentation	81
7.205.3.1 draw()	82
7.205.3.2 getTextHeight()	82
7.205.3.3 getTextWidth()	82
7.205.3.4 getType()	82
7.205.3.5 getWildcard1()	83
7.205.3.6 getWildcard2()	83
7.205.3.7 setWildcard1()	83
7.205.3.8 setWildcard2()	83
7.206TextAreaWithWildcardBase Class Reference	83
7.206.1 Detailed Description	84
7.206.2 Constructor & Destructor Documentation	84
7.206.2.1 TextAreaWithWildcardBase()	84
7.206.3 Member Function Documentation	84
7.206.3.1 calculateTextHeight()	84
7.207TextButtonStyle < T > Class Template Reference	85
7.207.1 Detailed Description	85
7.207.2 Member Function Documentation	86
7.207.2.1 setText()	86
7.207.2.2 setTextColors()	86
7.207.2.3 setTextPosition()	86
7.207.2.4 setTextRotation()	87
7.207.2.5 setTextX()	87
7.207.2.6 setTextXY()	87
7.207.2.7 setTextY()	87
7.208TextProgress Class Reference	88
7.208.1 Detailed Description	88
7.208.2 Constructor & Destructor Documentation	89
7.208.2.1 TextProgress()	89
7.208.2.2 ~TextProgress()	89
7.208.3 Member Function Documentation	89
7.208.3.1 getAlpha()	89
7.208.3.2 getColor()	89
7.208.3.3 getNumberOfDecimals()	89
7.208.3.4 getTypedText()	90
7.208.3.5 setAlpha()	90

CONTENTS

7.208.3.6 setColor()	890
7.208.3.7 setNumberOfDecimals()	891
7.208.3.8 setProgressIndicatorPosition()	891
7.208.3.9 setTypedText()	891
7.208.3.10setValue()	892
7.209TextProvider Class Reference	892
7.209.1 Detailed Description	892
7.209.2 Constructor & Destructor Documentation	893
7.209.2.1 TextProvider()	893
7.209.3 Member Function Documentation	893
7.209.3.1 getNextChar()	893
7.209.3.2 getNextLigature() [1/3]	893
7.209.3.3 getNextLigature() [2/3]	894
7.209.3.4 getNextLigature() [3/3]	894
7.209.3.5 initialize()	895
7.210Texts Class Reference	895
7.210.1 Detailed Description	896
7.210.2 Member Function Documentation	896
7.210.2.1 getLanguage()	896
7.210.2.2 getText()	896
7.210.2.3 setLanguage()	896
7.210.2.4 setTranslation()	897
7.211 TextureMapper Class Reference	897
7.211.1 Detailed Description	901
7.211.2 Member Enumeration Documentation	901
7.211.2.1 RenderingAlgorithm	901
7.211.3 Constructor & Destructor Documentation	901
7.211.3.1 ∼TextureMapper()	901
7.211.4 Member Function Documentation	901
7.211.4.1 applyTransformation()	901
7.211.4.2 draw()	902
7.211.4.3 drawTriangle()	902
7.211.4.4 getAlpha()	902
7.211.4.5 getBitmap()	903
7.211.4.6 getBitmapPositionX()	903
7.211.4.7 getBitmapPositionY()	903
7.211.4.8 getBoundingRect()	903
7.211.4.9 getCameraDistance()	903
7.211.4.10getCameraX()	904
7.211.4.11getCameraY()	904

lxxvi CONTENTS

7.211.4.12getOrigoX()		904
7.211.4.13getOrigoY()		904
7.211.4.14getOrigoZ()		904
7.211.4.15getRenderingAlgorithm()		905
7.211.4.16getScale()		905
7.211.4.17getSolidRect()		905
7.211.4.18getType()		905
7.211.4.19getX0()		906
7.211.4.20getX1()		906
7.211.4.21getX2()		906
7.211.4.22getX3()		906
7.211.4.23getXAngle()		906
7.211.4.24getY0()		907
7.211.4.25getY1()		907
7.211.4.26getY2()		907
7.211.4.27getY3()		907
7.211.4.28getYAngle()		907
7.211.4.29getZ0()		908
7.211.4.30getZ1()		908
7.211.4.31getZ2()		908
7.211.4.32getZ3()		908
7.211.4.33getZAngle()		908
7.211.4.34ookupRenderVariant()		909
7.211.4.35setAlpha()		909
7.211.4.36setBitmap()		909
7.211.4.37setBitmapPosition() [1/2]		909
7.211.4.3&setBitmapPosition() [2/2]		910
7.211.4.39setCamera()		910
7.211.4.40setCameraDistance()		
7.211.4.41setOrigo() [1/2]		910
7.211.4.42setOrigo() [2/2]		
7.211.4.43setRenderingAlgorithm()		911
7.211.4.44setScale()		911
7.211.4.45updateAngles()	٠.	911
7.211.4.46updateXAngle()		912
7.211.4.47updateYAngle()		912
7.211.4.4&updateZAngle()		912
7.212TextureSurface Struct Reference		912
7.213TiledImage Class Reference		
7.213.1 Detailed Description		914

CONTENTS

14
14
14
14
14
15
15
15
16
16
16
16
17
17
18
18
18
19
19
19
19
20
20
20
20
20
21
21
21
21
21
21 22
21 22 22
21 22 22 22
21 22 22 23
21 22 22 23 24
21 22 22 23 24 24
21 22 22 23 24 24
1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2

Ixxviii CONTENTS

7.217.2.1 TouchArea()
7.217.3 Member Function Documentation
7.217.3.1 draw()
7.217.3.2 getSolidRect()
7.217.3.3 getType()
7.217.3.4 handleClickEvent()
7.217.3.5 handleDragEvent()
7.217.3.6 setPressedAction()
7.218TouchButtonTrigger Class Reference
7.218.1 Detailed Description
7.218.2 Member Function Documentation
7.218.2.1 handleClickEvent()
7.219TouchCalibration Class Reference
7.219.1 Detailed Description
7.219.2 Member Function Documentation
7.219.2.1 setCalibrationMatrix()
7.219.2.2 translatePoint()
7.220TouchController Class Reference
7.220.1 Detailed Description
7.220.2 Constructor & Destructor Documentation
7.220.2.1 ~TouchController()
7.220.3 Member Function Documentation
7.220.3.1 init()
7.220.3.2 sampleTouch()
7.221 Transition Class Reference
7.221.1 Detailed Description
7.221.2 Constructor & Destructor Documentation
7.221.2.1 Transition()
7.221.2.2 ~Transition()
7.221.3 Member Function Documentation
7.221.3.1 handleTickEvent()
7.221.3.2 init()
7.221.3.3 isDone()
7.221.3.4 setScreenContainer()
7.221.3.5 tearDown()
7.222TwoWildcardTextButtonStyle < T > Class Template Reference
7.222.1 Detailed Description
7.222.2 Member Function Documentation
7.222.2.1 setTwoWildcardText()
7.222.2.2 setTwoWildcardTextColors()

CONTENTS

7.222.2.3 setTwoWildcardTextPosition()	. 934
7.222.2.4 setTwoWildcardTextRotation()	. 934
7.222.2.5 setTwoWildcardTextX()	. 934
7.222.2.6 setTwoWildcardTextXY()	. 935
7.222.2.7 setTwoWildcardTextY()	. 935
7.222.2.8 setWildcardTextBuffer1()	. 935
7.222.2.9 setWildcardTextBuffer2()	. 935
7.223TypedText Class Reference	. 935
7.223.1 Detailed Description	. 936
7.223.2 Constructor & Destructor Documentation	. 936
7.223.2.1 TypedText()	. 936
7.223.3 Member Function Documentation	. 937
7.223.3.1 getAlignment()	. 937
7.223.3.2 getFont()	. 937
7.223.3.3 getFontId()	. 937
7.223.3.4 getld()	. 937
7.223.3.5 getText()	. 938
7.223.3.6 getTextDirection()	. 938
7.223.3.7 hasValidId()	. 938
7.223.3.8 registerTexts()	. 938
7.223.3.9 registerTypedTextDatabase()	. 938
7.224TypedText::TypedTextData Struct Reference	. 939
7.224.1 Detailed Description	. 939
7.225UIEventListener Class Reference	. 939
7.225.1 Detailed Description	. 940
7.225.2 Constructor & Destructor Documentation	. 940
7.225.2.1 ~UIEventListener()	. 940
7.225.3 Member Function Documentation	. 940
7.225.3.1 handleClickEvent()	. 940
7.225.3.2 handleDragEvent()	. 940
7.225.3.3 handleGestureEvent()	. 940
7.225.3.4 handleKeyEvent()	. 941
7.225.3.5 handlePendingScreenTransition()	. 941
7.225.3.6 handleTickEvent()	. 941
7.226Unicode Class Reference	. 941
7,000 1 Marshay Type def Decymentation	
7.226.1 Member Typedef Documentation	. 942
7.226.1 Member Typedel Documentation	
	. 943
7.226.1.1 UnicodeChar	. 943 . 943

IXXX

7.226.2.3 itoa()	. 943
7.226.2.4 snprintf() [1/2]	. 944
7.226.2.5 snprintf() [2/2]	. 944
7.226.2.6 snprintfFloat() [1/2]	. 945
7.226.2.7 snprintfFloat() [2/2]	. 946
7.226.2.8 snprintfFloats() [1/2]	. 948
7.226.2.9 snprintfFloats() [2/2]	. 948
7.226.2.10strlen() [1/2]	. 950
7.226.2.11strlen() [2/2]	. 950
7.226.2.12strncmp()	. 950
7.226.2.13strncmp_ignore_white_spaces()	. 951
7.226.2.14strncpy() [1/2]	. 951
7.226.2.15strncpy() [2/2]	. 951
7.226.2.1@oUTF8()	. 952
7.226.2.17utoa()	. 952
7.226.2.18vsnprintf() [1/2]	. 953
7.226.2.19vsnprintf() [2/2]	. 953
7.227 Vector < T, capacity > Class Template Reference	. 954
7.227.1 Detailed Description	. 954
7.227.2 Constructor & Destructor Documentation	. 955
7.227.2.1 Vector()	. 955
7.227.3 Member Function Documentation	. 955
7.227.3.1 add()	. 955
7.227.3.2 clear()	. 955
7.227.3.3 contains()	. 955
7.227.3.4 isEmpty()	. 955
7.227.3.5 maxCapacity()	. 956
7.227.3.6 operator[]() [1/2]	. 956
7.227.3.7 operator[]() [2/2]	. 956
7.227.3.8 quickRemoveAt()	. 956
7.227.3.9 remove()	. 957
7.227.3.10removeAt()	. 957
7.227.3.11reverse()	. 957
7.227.3.12size()	. 957
7.228 Vector4 Class Reference	. 958
7.228.1 Detailed Description	. 958
7.228.2 Constructor & Destructor Documentation	. 958
7.228.2.1 Vector4() [1/2]	. 958
7.228.2.2 Vector4() [2/2]	. 958
7.228.3 Member Function Documentation	. 959

CONTENTS

7.228.3.1 crossProduct()	59
7.229 View < T > Class Template Reference	59
7.229.1 Detailed Description	59
7.229.2 Constructor & Destructor Documentation	30
7.229.2.1 View()	30
7.229.3 Member Function Documentation	60
7.229.3.1 bind()	60
7.230 Widget Class Reference	30
7.230.1 Detailed Description	31
7.230.2 Constructor & Destructor Documentation	31
7.230.2.1 Widget()	31
7.230.2.2 ~Widget()	31
7.230.3 Member Function Documentation	31
7.230.3.1 getLastChild()	31
7.230.3.2 getType()	32
$7.231 Wildcard Text Button Style < T > Class \ Template \ Reference \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	32
7.231.1 Detailed Description	63
7.231.2 Member Function Documentation	63
7.231.2.1 setWildcardText()	63
7.231.2.2 setWildcardTextBuffer()	64
7.231.2.3 setWildcardTextColors()	64
7.231.2.4 setWildcardTextPosition()	64
7.231.2.5 setWildcardTextRotation()	64
7.231.2.6 setWildcardTextX()	35
7.231.2.7 setWildcardTextXY()	35
7.231.2.8 setWildcardTextY()	65
7.232ZoomAnimationImage Class Reference	35
7.232.1 Detailed Description	38
7.232.2 Member Enumeration Documentation	38
7.232.2.1 States	38
7.232.3 Constructor & Destructor Documentation	39
7.232.3.1 ZoomAnimationImage()	39
7.232.3.2 ~ZoomAnimationImage()	39
7.232.4 Member Function Documentation	39
7.232.4.1 getAlpha()	39
7.232.4.2 getAnimationDelay()	39
7.232.4.3 getLargeBitmap()	39
7.232.4.4 getScalingMode()	70
7.232.4.5 getSmallBitmap()	70
7.232.4.6 getType()	70

Ixxxii CONTENTS

7.232.4.7 handleTickEvent()	970
7.232.4.8 isRunning()	970
7.232.4.9 isZoomAnimationRunning()	971
7.232.4.10setAlpha()	971
7.232.4.11setAnimationDelay()	971
7.232.4.12setAnimationEndedCallback()	971
7.232.4.13setBitmaps()	972
7.232.4.14setCurrentState()	972
7.232.4.15setDimension()	972
7.232.4.16setHeight()	972
7.232.4.17setPosition()	973
7.232.4.18setScalingMode()	973
7.232.4.19setWidth()	973
7.232.4.20startTimerAndSetParameters()	974
7.232.4.21startZoomAndMoveAnimation()	974
7.232.4.2&startZoomAnimation()	975
7.232.4.23updateRenderingMethod()	975
7.232.4.24updateZoomAnimationDeltaXY()	976
	077

Index

Chapter 1

Changelog

4.12.3

* Release date: September 25rd, 2019

- * New TouchGFX Core Features (since 4.12.0):
 - * Binary Fonts: Binary fonts can be used as an alternative to compiling and linking font information in to your application. The main advantages of this approach is to get a smaller application binary and get a flexibility in providing different sets of fonts with your device.
 - * Font Caching: Support for caching binary fonts, suitable for loading only the required characters from a file system, when a string is displayed.
 - * Binary Translations: Support for binary translations, suitable for loading translations from a file system as opposed to linking them into the application.

Read more about these feature here:

https://touchgfx.zendesk.com/hc/en-us/articles/360024979552

- * Support for non-memory-mapped flash storage for 16bpp displays, allows storage of images and fonts in e.g. inexpensive SPI flashes.
- * Recognition of Unicode sequences for Arabic ligatures Allah,
 Akbar, Mohammad, Salam, Rasoul, Alayhe, Wasallam and Rial Sign.
- * Bugfixes in TouchGFX Core:
 - \star TextureMapper (bilinear) would fail to draw L8_RGB888 and RGB888 bitmaps on 24bpp displays correctly.
 - \star Setting a text with a wildcard in a TextArea without wildcard support in combination with RTL could cause a crash.
 - * If a CacheableContainer was smaller than the associated bitmap, the size of the container would not be correct.
 - * Fixed SnapshotWidget on 8bpp LCDs.
 - * Fixed rendering of some Arabic ligatures.
 - * Fixed rendering of some Hindi ligatures.
 - \star Fixed bug when applying certain GSUB substitution rules.
 - * Fixed bug that binary fonts contained extra rules.

4.12.2

- * Release date: August 22nd, 2019
- * New TouchGFX Core Features:
 - * WordWrapping wide text using TextArea::setWideTextAction() now wraps at normal space as well as Unicode characters 0x200B (Zero Width Space).
- * Bugfixes in TouchGFX Core:
 - * Binary fonts: The fontConverter tool was not writing kerning data into binary font files when the "binary_fonts" option was specified in the application configuration. This caused texts to

appear incorrect when using binary fonts.

4.12.1 _____

- * Release date: August 15th, 2019
- * Updated "Third Party Components.pdf" to reflect updated components
 - * libpng-1.6.36
 - * zlib-1.2.11
 - * freetype-2.9.1
- * Bugfixes in TouchGFX Designer:
 - \star Fixed a bug where having a delay action together with a button clicked action would result in compilation errors.
 - * Fixed a bug where Canvas Buffer for a Screen was not correctly updated when adding a Canvas Widget to a Custom Container Instance.
 - * Fixed a bug where an error message in the Online Applications window would get stuck.
 - \star Fixed faulty rendering on the Canvas in the Designer when using the Alpha value of the different Progress Indicators.
 - \star Fixed a bug where creating a new project, closing the Designer without saving it, and reloading the project would cause the project to have no available typographies.
 - \star Updated error message when trying to import an already open UI into another project to be more clear.
 - * Fixed a bug where the Text Manager could have multiple foci visually in a specific circumstance.
 - * Fixed a bug where the Properties tab for a Widget would not display a red border correctly, when an error is present on the Widget.
 - \star Fixed a bug where using the Consolas font would render incorrectly on the Canvas in the Designer after reloading a project using that font.
- * Bugfixes in TouchGFX Core:
 - * TextureMapper bug if Display Rotation was in use.
 - * Disregard kerning data for CachedFont.
 - * CachedFont did not look in font cache for the fallback character.

4.12.0

- * Release date: 06-07-2019
- * Important upgrade information:
 - * Public version of drawGlyph has been removed. Use drawString instead.
 - \star Using bitmap format ARGB8888 for opaque images will no longer dither to 565 but keep full 888 colors. Using ARGB8888 for non-opaque images will still dither to 565 when the opaque format is RGB565.
 - * Images converted to BW_RLE will no longer fall back to BW if the BW RLE format causes the converted image to be larger. Instead a warning will be generated by the image converter. Use the Designer (or the new configuration file) to specify ${\tt BW}$ or ${\tt BW_RLE}$ for each individual image.
- * New TouchGFX Designer Features:
 - * A custom container can now be nested within another custom container. This enables composing custom components into larger custom components indefinitely.
 - \star A custom container supports defining custom triggers and custom actions, a screen supports defining custom actions. These triggers and actions support the flow of information from one component to another component. Using such triggers and actions in interactions within the Designer enables doing more real world application behaviour without leaving the Designer. Check out the documentation for further introduction.
 - \star A Container can now be generated as a CacheableContainer.
 - * A new "Images" tab has been added for setting up individual image

- configurations (Image Format, Dither Algorithm, Layout Rotation, etc.).
- \star Application settings and other new settings have been relocated to the "Config" tab.
- * New TouchGFX Core Features:
 - \star Upgraded 3rd party libraries used by framework tools. This results in much nicer looking texts.
 - * Improved kerning through larger kerning table.
 - * Thai fonts are now rendered better with tighter line spacing and better rendering of Sara Am in some cases.
 - * Preliminary support for Hindi (Devanagari). The following GSUB tables are applied: nukt (Nukta Forms), akhn (Akhands), rkrf (Rakar Forms), cjct (Conjunct Forms), vatu (Vattu Variants), rphf (Reph Forms), pref (Pre-Base Forms), half (Half Forms), blwf (Below-base Forms), abvf (Above-base Forms), pstf (Post-base Forms), and cfar (Conjunct Form After Ro). The following are NOT currently supported: abvs (Above-base Substitutions), blws (Below-base Substitutions), and psts (Post-base Substitutions). Also, not all GSUB tables types are supported.
 - * Added a new Line::updateLengthAndAngle() API.
 - * Added support for partial framebuffers rendering and updates.
 - * Added simple string printing for debugging.
 - * Allow changing the BitmapCache after initialization.
 - * New macros for setting sections names for flash programming.
 - * Added Circle::updateArc() to update arc start and arc end with minimal invalidation areas.
 - * Updated CircleProgress to use higher precision calculations for updates.
 - * Added CacheableContainer for offscreen widgets rendering.
 - \star Added support for L8 graphics assets with 16bit, 24bit and 32bit palettes.
 - * Added support for L8 hardware acceleration via DMA2D.
 - * Added new LCD32bpp framebuffer renderer.
- * Bugfixes in TouchGFX Designer:
 - * ProgressIndicator is updated automatically after call to CircleProgress::setStartEndAngle(),
 ImageProgress::setAnchorAtZero() and
 TextProgress::setNumberOfDecimals().
- * Bugfixes in TouchGFX Core:
 - \star Fixed redraw of circleProgressIndicator when setting new value.
 - * Removed additional screen redraw after a screen transition is complete. This additional redraw caused performance issues on some platforms. Invalidating the entire screen in Screen::afterTransition(), if required, is now the responsibility of the application developer.

4.11.0

- * Release date: March 1st, 2019
- * Important upgrade information:
 - \star If your application includes LCD.hpp and expects to have access to HAL, this will no longer work since LCD.hpp no longer includes HAL.hpp. Make sure to include HAL.hpp in this case. Older versions of sample applications <code>Demo1</code> and <code>Demo2</code> had this issue and have been updated.
- * New TouchGFX Designer Features:
 - \star Added Bring Forward/Send Backwards support for widgets, via UI Buttons and keyboard shortcuts Ctrl + F, Ctrl + B.
 - * Added support for copy and paste of Screens and CustomContainerDefinitons.
 - \star Added support for reordering CustomContainerDefinitions.
 - * Switching between Screen and CustomContainerDefinitions now remembers the previously selected Screen and CustomContainerDefinition.
 - \star The last used typography is now used when creating new texts and widgets that use text.

- * Added new tree icon for CustomContainerInstances.
- * Disabled continuous code generation and compiling.
- * Improved readability of the output in Detailed Log window.
- * Widget Wildcard Characters added to the Texts tab, which adds default wildcard characters when using some widgets
- * Improved performance when loading a project.
- * Improved performance when generating a project.
- * Improved performance of validation engine.
- * Added support for 6 bit color displays (8bpp).
- \star Added support for setting RadioButtonGroup for RadioButtons.
- * Added support for Display Rotation (Landscape/Portrait).
- \star Added support for setting Landscape/Portrait simulator skins in the Designer.
- * Added support for the following widgets: AnalogClock, DigitalClock, TextureMapper, AnimatedTextureMapper & Shape.
- * The Designer now generates the Makefile and Visual Studio files used for running the Simulator.
- * New TouchGFX Core Features:
 - * Added support for 6 bit color displays (RGBA2222, BGRA2222, ARGB2222 and ABGR2222 framebuffer formats).
 - * Added support for Thai.
 - * Improved rendering of Arabic text.
 - * Added handling of negative line spacing.
- * Bugfixes in TouchGFX Designer:
 - * Fixed Ctrl + A (select all) not working for CustomContainerDefinitions.
 - \star Fixed reordering of Screens selecting the first screen in the list and deleting the undo/redo history for the Screen that was moved.
 - \star Fixed bug where the undo/redo history would become broken after selecting the Application node.
 - * Fixed application names not being allowed to start with a number or contain "-" or "_".
 - \star Fixed loading an application while on the CustomContainer tab resulting in erroneous content on the canvas.
 - * Fixed pressing undo after moving multiple elements into a container resulting in a crash.
 - * Fixed font files being locked when loading a project.
 - * Fixed error not showing up on components that use text, when removing their Resource Text.
 - \star Fixed bug where loading a faulty application by double-clicking a TouchGFX file would cause the splash screen to get stuck.
 - * Fixed faulty position code generation for ModalWindow.
 - * Fixed missing "Move widget" interaction support for ScrollableContainer, ScrollList & ScrollWheel.
 - * Fixed the ordering of the Recent Applications list. Now correctly updates when opening an application.
 - \star Fixed bug where inserting a widget could add an empty undo item to the undo/redo history.
 - \star Fixed missing header text and description in the properties pane for CustomContainerDefinitions.
 - * Fixed bug where idle CPU usage was higher than expected.
 - * Fixed bug where setting an interaction on a FlexButton inside a CustomContainer would generate faulty code.
 - * Fixed bug where setting a mixin on a widget was not undo-able.
 - \star Fixed missing undo/redo functionality for adding styles to FlexButton.
 - \star Fixed wrong order of initializations when using numerous slider callbacks in interactions.
- * Bugfixes in TouchGFX Core:
 - * Fixed precision in CWR Painters for 4bpp and 2bpp.
 - \star Fixed precision in alpha blending formulaes for 8bpp, 16bpp and 24bpp.

4.10.0

=====

- * Release date: November 5th, 2018
- * Requirements:

- * TouchGFX is now only available for STM32 microcontrollers.
- * New TouchGFX Designer Features:
 - * Added support for the following widgets: ImageProgress, BoxProgress, TextProgress, LineProgress, CircleProgress, Line, Circle, BoxWithBorder, FlexButton, ScrollList, ScrollWheel and SwipeContainer.
 - * Canvas Buffer setting can be adjusted on screens.
 - * Support for screen transition: CoverTransition.
 - * Now logs the following system information for use in support scenarios: Username, Designer version, Designer installation path, Windows version, Current culture, Installed .NET versions.
 - * It is now possible to import a UI with any resolution to an application (resolution check has been removed).
 - * Added button to show/hide clipped widgets.
 - * Improved performance when dragging and resizing widgets on the canvas.
- * New TouchGFX Core Features:
 - \star Circle and AbstractShape now supports higher precision on arc start and arc end for smoother arcs.
 - * The internal Q5 structure now uses 32 bit instead of 16 bits for increased value range.
 - * Added Circle::getPrecision().
 - * Added functions FadeAnimator::isFadeAnimationRunning(),

MoveAnimator::isMoveAnimationRunning(),

AnimatedImage::isAnimatedImageRunning() and

 ${\tt ZoomAnimationImage::isZoomAnimationImageRunning().} \ \, {\tt The old isRunning()} \ \, {\tt functions have been deprecated.}$

- * ListLayout::setDirection() and getDirection() added.
- * Updated roo gem from 1.13.1 to 2.7.1.
- \star Pressing SHIFT-F3 will copy the screen to the clipboard (Windows only).
- \star Pressing CTRL-F3 will save the next 50 screens to the screenshots folder.
- * Generated assets are now indented properly.
- * ScrollableContainer::setScrollbarsPermanentlyVisible() added.
- * Bugfixes in TouchGFX Designer
 - * Fixed ModalWindow widget not resizing when Screen or Custom Container size changes.
 - * Fixed generating code failing if a files hidden attribute was set to hidden.
 - \star Fixed changing the casing of a screen or custom container name resulting in a recompilation error.
 - * Fixed bug where internet loss would crash the Designer if no Online Applications are available.
 - \star Fixed ModalWindow widget position being generated incorrectly after loading a project.
 - * Removed unnecessary recompilation when loading Designer project.
 - * Fixed visual bug in ImagePicker where the "empty placeholder" would show up even though you have subfolders in current folder.
 - * Fixed bug where the Designer was not using default credentials through proxy server.
 - * Fixed bug where the Designer would not correctly report an error when trying to flash to a wrong target.
 - \star Fixed bug where having insufficient permissions to write to the chosen touchgfx path would crash the Designer.
 - * Fixed bug where the Designer was incorrectly interpreting screen changes as an unsaved change.
 - * Fixed a visual bug, where widgets inside a Container would not display properly when resizing the Container.
 - * The Designer now closes a running Simulator process, when you load another application.
 - \star Fixed a bug where it was possible to drag widgets inside an instance of a Custom Container.
 - * Circle did sometimes not render correctly, and invalidated rectangle was not calculated properly.
 - \star Fixed Circle when half line width was greater than radius.
- * Bugfixes in TouchGFX Core:
 - \star Fixed erroneous calculation of x & y values in setValue in LineProgress.cpp.

- \star Circle did sometimes not render correctly, and invalidated rectangle was not calculated properly.
- * Fixed Circle when half line width was greater than radius.
- \star Fixed drawing lines longer than 2047 pixels, e.g. 1449 pixels wide and 1449 pixels high.
- * Fixed bug preventing some Arabic ligatures from being rendered correctly.

4.9.4

* Release date: January 25th, 2018

- * Bugfixes:
 - * Reduced the time it takes to load an application in the Designer.

4.9.3

- * Release date: December 15th, 2017
- * Buafixes:
 - * Designer now uses default Windows proxy settings.
 - * Package manager updates available packages when online.
 - * Improved error description when offline.
 - * Set text interaction works with resource texts.
 - \star Project updater updates MSVS projects with correct image formats.
 - \star Text size calculated wrongly in Designer in rare occasions.
 - \star Recent files ordered by date.
 - \star Corrected initialization of counter in Wait For interaction.
 - * Fixed drawing of child elements in list layout, when resized.
 - * Fixed loading of application with list layout widgets.
 - * .otf font files now correctly rendered.
 - * Dragging containers could in rare cases introduce wrong coordinates.
 - \star Fixed zero termination of wildcard text buffers.
 - \star Button With Label text rendering correction.
 - \star tgfx.exe packager works for more complex file layouts.
 - * Source code included for containers.
 - * Additional minor Designer UI fixes and improvements.

4.9.2

- * Release date: November 20th, 2017
- * Bugfixes:
 - \star Fixed Designer issue where dragging elements on the canvas would in some cases cause an exception.

4.9.1

- * Release date: November 16th, 2017
- * Bugfixes:
 - * Fixed several Designer issues with TextArea widgets when placed inside containers.
 - * Fixed an issue with interactions triggered by "Another interaction is done" dissappearing when loading a project.
 - * On PCs with certain security policy configurations, the Designer was not able to create new projects correctly.
 - \star Improved error handling in Designer if the asset generation, code compilation or post generation commands fail.
 - * Fixed an issue where the TouchgfxPath in Designer project files was not interpreted correctly.
 - * Some typography changes in Designer did not cause new code to be generated.
 - \star Fixed issue with ImageConverter when assets folder was under svn control.

* ImageConverter could in certain cases fail to detect changes in assets.

4.9.0

* Release date: November 8th, 2017

* New Features:

- * Added a package manager for handling board support packages, demos and examples. The Designer will now fetch these from an online repository. Read more about this feature here: https://touchgfx.zendesk.com/hc/en-us/articles/115002589211
- * All the old examples, demos and ports for various boards have been removed from the framework, and are now available as packages instead
- * Substantially improved text handling in the Designer. It is now possible to work with translations and wildcards in the Designer, so it should no longer be necessary to edit the texts.xlsx file manually.
- * Designer is now much more flexible regarding application file structure, and is now able to auto-update IAR and Keil IDE projects regardless of file location.
- * Added Designer support for the ScrollableContainer and ListLayout widgets.
- * Added support for the SW4STM32 IDE.
- \star Added support for version 8.10 of IAR Embedded Workbench.
- * Image converter now has an option to operate on folders, instead of being invoked once per .png file. This substantially speeds up the process of converting images. This mode is the default behavior for new projects.
- * The GNU Arm Embedded toolchain (GCC cross-compiler) has been updated to version 6-2017-q2-update (gcc version 6.3.1).
- \star The GNU compiler for the PC simulator has been updated to version 6.3.0.
- * Added gcc core libs compiled with -mfloat-abi=hard for Cortex-M4f and Cortex-M7.
- * Increased number of widgets that can be registered as timer widgets from 24 to 32. Also added functions for obtaining information about which widgets are currently registered.

* Bugfixes:

- * AnimationTextureMapper::cancelMoveAnimation() is renamed to cancelAnimationTextureMapperAnimation() to avoid problems with MoveAnimator::cancelMoveAnimation().
- \star Fixed bug in PainterRGB565Bitmap when rendering solid pixels from an ARGB8888 Bitmap.
- * Fixed rare bug in FontConvert if all used characters are missing from the font.
- \star Fixed unitialized variables in the DMA class.

* Update procedure:

* For this release additional steps might be needed. Please refer to the Known Issues article for details: https://touchgfx.zendesk.com/hc/en-us/articles/207507415

4.8.0

* Release date: March 10th, 2017

* Performance

- * LCD4bpp now draws characters up to 15% faster.
- * Canvas widgets now render slightly faster in certain situations.

* New Features:

- * TouchGFX Designer released. The core framework, Designer and environment shell are now bundled in a single installation. To get started with the Designer, please see https://touchgfx.zendesk.com/hc/en-us/articles/115001801745
- \star Support for Farsi and Arabic ligatures where squences of up to

- three character are recognized.
- * Added support for Microsoft Visual Studio 2017.
- * TextArea and TextAreaWithWildcard(s) now support setWideTextAction() to automatically break lines and insert ellipsis at end of line, when the line is too long.
- * Added getter functions to Slider.
- * MoveAnimator and FadeAnimator can now clear the callback set for animation ended.
- * Errors from ImageConvert, TextConvert and FontConvert are now shown in the Error List window of Visual Studio.
- * Simulator applications are now Windows programs instead of Console programs. To use printf() or std::out, please see https://touchgfx.zendesk.com/hc/en-us/articles/205074511-Tips-tricks

* Bugfixes

- * AbstractShape::updateAbstractShapeCache() is now a public function and should be called after one or more calls to AbstractShape::setCorner(), to ensure shape is correct.
- * Simulator window can no longer be unintentionally resized.
- * F2 to highlight invalidated areas now works with old HALSDL.
- * PainterGRAY2Bitmap, PainterGRAY4Bitmap, PainterRGB565Bitmap and PainterRGB888Bitmap all failed to validate that painting was inside the size of the bitmap in some situations.
- * HALSDL2 (simulator) now uses 24bpp on screen to make colors in screenshots correct.
- * TiledImage::setOffset() now handles an empty bitmap correctly.
- * TiledImage::getSolidRect() would sometimes report wrong rect.
- * If text in a TextArea was rotated, resizeToCurrentText() and resizeHeightToCurrentText() would swap the width/height.
- * Function getTextHeight() would not take line spacing into account. Functions like resizeToCurrentText() and others that use the getTextHeight() function would not resize correctly.
- * SlideMenu::setState() did not handle EXPANDED state correctly.

* Update Procedure

- * Due to the addition of TouchGFX Designer, installation is now done via an .msi installer. For details and manual installation refer to https://touchgfx.zendesk.com/hc/en-us/articles/206819819
- * Compatible with existing 4.x applications and HAL ports.

4.7.0

- * Release date: December 14th, 2016
- * New Features:
 - * Source code for all the standard widgets and containers is now included. See the touchgfx/framework/source/touchgfx directory. Note that these classes are still present in the core library, and the source code files are not added to the IAR/Keil/gcc projects per default. For details, see the following article: https://touchgfx.zendesk.com/hc/en-us/articles/115000035825
 - * Optimized the handling of single frame buffer configuration on TFT controller based platforms, which in many cases eliminate the need for external RAM. For details on this feature, and how to enable it please see the following article:
 - https://touchgfx.zendesk.com/hc/en-us/articles/203649441
 - * Substantial performance optimizations of the canvas widget system and all the standard painters. Expect a very significant increase in performance if many pixels are being drawn, and a smaller increase in performance for minor shapes (e.g. graph lines). The "PainterVerticalAlpha" used in our demos have also been updated. If you are using custom painters please refer to the Known Issues article:
 - https://touchgfx.zendesk.com/hc/en-us/articles/207507415
 - * The text converter tool will now combine identical translations across all languages, resulting in reduced footprint. The result of this process will be printed during asset generation.

 NOTE: This behavior is enabled by default. If you have an existing project where you manipulate the text data structures (e.g. load a single language into RAM), this optimization might break your code. The optimization can be disabled by adding the following

remap_identical_texts := no (for "make"-based generation)
<RemapIdenticalTexts>no</RemapIdenticalTexts> (for MSVS)

- * Updated SDL version used by simulator from 1.2 to 2.0.4. SDL1.2 is still present in the distribution, but all examples and projects now use SDL2. For more details, please see: https://touchgfx.zendesk.com/hc/en-us/articles/115000011245
- * Support for skinning the simulator with .png files. If the .png files contain non-opaque areas, the simulator window will be shaped accordingly. See display_orientation_example for a code example or read the following article:

https://touchgfx.zendesk.com/hc/en-us/articles/115000014669

- \star On ST targets with Chrom-ART, the Box widget will now be drawn by DMA even when alpha < 255 (BLIT_OP_FILL_WITH_ALPHA support).
- * TextArea and TextArea with wildcard(s) now support setWideTextAction() to automatically wrap long lines.
- * Added the ability to display a "fallback" character in case a non-existing glyph is encountered at runtime. This is configured in the typography sheet of the text database.
- * Added options for forcing the inclusion of additional glyphs in a font. This makes it much easier to handle dynamic texts where the glyphs are not known at compile time. This is configured in the typography sheet of the text database.
- * Output from the TextConvert utility is now post-processed to give significant saving by mapping identical strings to the same memory areas.
- * Added built-in BitmapId called BITMAP_ANIMATION_STORAGE which can be used to refer to the animation storage when assigning a Bitmap to a widget.
- * Added dither algorithm selection from config/gcc/app.mk and config/msvs/Application.props.
- \star It is possible to save a simulator screenshot programatically, by using: # ifdef SIMULATOR

(static_cast<HALSDL2*>(HAL::getInstance()))->saveScreenshot();
#endif

- * ScrollableContainer now properly ignores invisible elements.
- * DigitalClock now supports a zero to be displayed in front of the hour indicator (if hour < 10).
- \star The simulator can now highlight the areas being invalidated. Press F2 to toggle this feature.
- \star Added Unicode::vsnprintf functions that take va_list arguments instead of ellipsis.

* Bugfixes

- * Unicode::sprintfFloat did not print <space> instead of '+' if the format string was "% f". Also, the sign of floating point numbers in range]-1..0[would not be printed with sign so for example -0.5 would print as 0.5.
- \star Fixed a bug that could cause TextureMapper to read outside source bitmap memory area.
- * GPIO.cpp for STM32F769-Discovery and Eval boards had some incorrect GPIO pin manipulations (used for performance measurement).
- * Some methods in Slider.hpp were missing a virtual declaration.
- * Fixed a bug in BoardConfiguration for STM32F769-Discovery board causing 24bpp color mode to be displayed incorrectly.
- * AnimatedImage setBitmap(..) should not be used and is now private For AnimatedImage use setBitmaps(..) instead.
- * Project files and Makefile have been updated to allow the TouchGFX framework to be placed on another disk drive than the project being developed.
- * TouchGFX Environment (version 2.8)
 - * "make.exe" is now version 4.1 which allows for parallel compilation, by adding e.g. "-j8" to your make command. This substantially speeds up compilation. If your makefile is from TouchGFX 4.2.0 or earlier, you will need to either update it, or to use make-3.81.exe
 - * g++ could in some cases report "There is no disk in the drive. Please insert a disk into drive E:.". This has been fixed by upgrading gcc from version 4.8.1 to version 4.9.3.

- * Release date: September 12th, 2016
- * Performance
 - * Optimization improvements of core library for GCC on Cortex-M4 and Cortex-M7, providing significant speedup of especially TextureMapper and Canvas widgets compared to TouchGFX 4.6.0.
- * New Features
 - * New function in HALSDL to set title of simulator window see HALSDL::setWindowTitle().
 - * BW_RLE format (1bpp displays) now compresses better. Remember to remove old generated files and re-generate assets.
 - \star STM32F756G-EVAL using IAR now supports flashing of external memory.
- * Bugfixes
 - * Added IAR linker redirect commands to fix linker errors when compiling a Cortex-M4 based target with IAR 7.x.
 - * Assigning different memory buffers to CanvasWidgetRenderer using setupBuffer() could in rare cases result in memory corruption.
 - * TextureMapper could in rare cases draw outside the frame buffer.
 - * Setting the offset of a TiledImage did not work properly.
 - \star Fixed two issues that would in some cases cause memory corruption when deleting dynamic bitmaps.
 - * Missing virtual method declarations in AnalogClock added.
 - * Fixed a problem in GCC linker script for LPC4088DisplayModule which caused texts and fonts to be placed in external flash.
 - * For those using fontconvert.out on its own, the output directory is now automatically created if it does not exist.
 - * ScrollableContainers could in rare cases send a wrong drag event to a child.
 - * Monochrome (1bpp) displays with width not divisible by 8 would not display text correctly.
 - \star Slightly increased default touch sample rate on STM32F746G Discovery board.

4.6.0

- \star Release date: June 14th, 2016
- * New features
 - * Added support for 2bpp grayscale displays. See https://touchgfx.zendesk.com/hc/en-us/articles/208237889 for details.
 - * Added support for 4bpp grayscale displays. See https://touchgfx.zendesk.com/hc/en-us/articles/209003105 for details.
 - https://touchgfx.zendesk.com/hc/en-us/articles/209003105 for details * New widget TiledImage. Will display one or more repetions of an
 - image. The number of repetitions depends on the size of the widget and the size of the image.
 - * New widget RepeatButton. A button that will repeatedly fire click events when pressed.
 - * New widget AnimationTextureMapper. TextureMapper with build in animation features. See animation_texture_mapper_example.
 - * New containers AnalogClock and DigitalClock, see clock_example.
 - * New containers ProgressIndicators, see progress_indicator_example.
 - * New container ModalWindow. Creates a window on top of the main screen and a shade on the rest of the main screen. No clicks are passed on to the main screen as long as the modal window is visible. See example modal_window_example.
 - * New container SlideMenu. Animating side/top/bottom-menu that has an activate button for sliding it in/out of the screen. A timeout can be set for automitical hiding when idle for a period of time.
 - \star Canvas Widget Line supports ROUND_CAP_ENDING and setCapPrecision() to control the round cap.
 - \star Simulator can now generate ticks very close to the frequency of the hardware. See
 - https://touchgfx.zendesk.com/hc/en-us/articles/205074511 for details.
 - * Mouse X and Y coordinates are put in the title of the window in the simulator. (press F1 to (de)active this when running the simulator). See

https://touchgfx.zendesk.com/hc/en-us/articles/205074511 for details.

- * ST Cube drivers updated to version 1.4.0 for STM32F7 MCU and STM32F7 based boards.
- * Added support for the STM32769I-EVAL board.
- * Added support for the STM32F769I-Discovery board.
- * Screenshots made from the simulator (F3) are now saved under a name with timestamp to prevent old screenshots to be overwritten by accident.
- * Simulator now outputs canvas widget memory usage to easily find optimal canvas memory buffer size. See https://touchgfx.zendesk.com/hc/en-us/articles/205074511 for details.

* Bugfixes

- * DMA drivers for ST boards: express DMA2D instance initialization for STM32F7. Fixed incorrect used of CLUT_CM for F4-Discovery.
- * DMA drivers for LPC17xx, LPC18xx, LPC43xx did not behave correctly if other DMA channels are in use simulatenously. They now properly look at flags for channel 0 only.
- * Touch controller drivers for ST boards now properly checks that initialization was OK before querying.
- \star Mouse clicks in the simulator would not always be detected.
- * ImageConvert.exe has RGB565 as default (and sensible defaults for other opque formats)
- * ImageConvert would not work for a BW image scheduled for compression (BW_RLE) and rotation (.90. in filename) if the image would become too large if compressed (falling back to BW format).
- * All Makefiles now use abspath instead of realpath.
- \star AnimatedImage now allows the animation to be restarted from the AnimationEnded callback function.
- \star QSPI flash size corrected to 64MBytes for STM32756G-EVAL board.
- * Added D-cache invalidation to STM32F7HAL::flushFrameBuffer. This fixes occasional graphics errors on STM32F7 when in single frame buffer mode and fb was located in SRAM.
- * The otm8009a displays (STM32769-DISCO, STM32769-EVAL, STM32469-DISCO, STM32469-EVAL) are now using maximum display brightness.
- * Added a workaround for a bug in IAR 7.50.x regarding va_list name mangling.
- * Update Procedure
 - * Compatible with existing 4.x applications and HAL ports. Please refer to this article for details: https://touchgfx.zendesk.com/hc/en-us/articles/206819819

4.5.1

- * Release date: March 14th, 2016
- * Bugfixes
 - * Fixed two IAR linker issues related to resolving the va_list symbol, which would cause some versions of IAR being unable to link the example projects.
 - * STM32F4-Discovery board would draw solid rectangles with the wrong color in 16bpp mode.
 - * The Canvas Widget Renderer no longer performs unaligned memory accesses.
 - * vApplicationIdleHook (FreeRTOS specific) no longer blocks, which previously prevented FreeRTOS from freeing memory if tasks were deleted.
 - \star Arabic words with accent in the middle would not render properly.
 - * Added PixelDataWidget::getAlpha().
 - * Unicode::strncpy() with a char* as source would not copy characters with ascii codes above 127 properly.

4.5.0

- * Release date: February 2nd, 2016
- * New features
 - * Support for two new languages, Arabic and Hebrew, with right-to-left text rendering. RTL strings can be mixed with LTR

- texts and numbers.
- * Support for 24 bits per pixel frame buffers. Images look more detailed, but also consume more memory. See this article: https://touchgfx.zendesk.com/hc/en-us/articles/206725849
- * Bitmaps can now be created at runtime using method Bitmap::dynamicBitmapCreate. Useful for e.g. displaying .bmp files loaded from an SD card. See dynamic_bitmap_example and this article: https://touchgfx.zendesk.com/hc/en-us/articles/207460605
- * Frame rate compensation feature which provides smoother animations if frame rate occasionally drops. Not enabled by default. See article: https://touchgfx.zendesk.com/hc/en-us/articles/206430529
- * Bitmap caching is enhanced to allow removal of bitmaps from the cache to make room for caching of other bitmaps. See this article: https://touchgfx.zendesk.com/hc/en-us/articles/205953932
- * A new widget, PixelDataWidget, is introduced. This widget makes it possible to display raw pixel data obtained at runtime (e.g. video samples).
- \star The simulator executable on windows now features an icon for easier identification in the task bar.
- * ST boards supported by TouchGFX can now have just their internal flash programmed from the command using 'make intflash' provided that ST-Link Utility Release 3.7+ is installed. For usage and troubleshooting, please refer to this article: https://touchgfx.zendesk.com/hc/en-us/articles/205264831
- * Unicode::snprintf() has been improved and updated substantially to support more of the standard format specifiers like %02d.
- * Unicode::snprintfFloat() added to support floats (in separate function because the "%f" va_args approach would force inclusion of doubles).
- * Quality of image converter dithering has been improved (floating point arithmetics). Also added support for new types of dither algorithms, and can take into account hardware with various wiring of the low (unused) bits in 16/18 bit displays.
- * touchgfx::ButtonWithLabel now contains a method, updateTextPosition(), that can be used to ensure horizontal text centering when changing label content (e.g. when changing language).
- * touchgfx::TextArea has a new method, setBaselineY(), that allows placing texts according to a text baseline instead of upper left corner.
- * The internal format of glyph encoding now stores the first pixel in the least significant bit instead of the most significant bit.
- * Specification of color values has been switched from uint16_t to colortype to support seamless switching between 16 and 24 bit colors.
- * The touchgfx::TextArea class now has a method, setIndentation(), that can prevent the glyph of characters from being cut off in the rare case where it extends under the previous character (similarly for touchgfx::Keyboard class which has a new setTextIndentation() method).
- * STM32F7xx and STM32F4x9 ports now support DMA transfers of touchgfx::Box.
- * The GPIO::VSYNC_FREQ signal was previously "toggled" exclusively on "VSYNC" interrupt (NXP LPC18xx, NXP LPC43xx, Freescale MK70F12, ST stm32f4x9). The signal is now high on "VSYNC" interrupt and low on "Front-Porch-Entered" interrupt.
- * GCC support for Cortex-M3.

* Bugfixes

- \star Fixed rare crash on STM32F7 caused by speculative caching of invalid QSPI memory region. Update your BoardConfiguration if yours is based on 4.4.x.
- * Fixed occasional display flickering on STM32F746G-DISCO board caused by cache access on FMC bank 1.
- * Handling of the character "%"" in touchgfx::TextAreaWithWildcards has been improved to prevent inserting %% in some special cases.
- * touchgfx::DragEvent and touchgfx::GestureEvent now use and report signed coordinates instead of unsigned. This makes more sense as drags/gestures are expressed in coordinates relative to the drawable receiving them.
- * snprintf("%x") would generate uppper case hex. Now "%X" generates uppercase hex and "%x" generates lower case hex, just like the standard snprintf().

- * Fixed randomness for demos when running on Linux.
- \star Fixed redrawing when using heavily italicized fonts.
- * Pointer to ModelListener in Model class for all TouchGFX applications was not properly initialized (NULL).
- * Fixed support for heavily italicized fonts in touchgfx::TextArea.
- * Subtle error in the Image Converter where column 0 could get slightly incorrect pixel colors. As a result the entire image could be slightly wrong, probably not noticeable.
- * Minor error in Slider where values were not distributed evenly.
- * Deprecated
 - * LCD::drawGlyph() has been deprecated. Use LCD::drawString instead.
- Update Procedure
 - * Compatible with existing 4.x applications and HAL ports. Please refer to this article for details: https://touchgfx.zendesk.com/hc/en-us/articles/206819819

4.4.2

- * Release date: November 26th, 2015
- * Bugfixes:
 - * Corrected rare GUI task hangup on STM32F7 targets when compiling with IAR 7.x $\,$

4.4.1

- * Release date: October 27th, 2015
- * Bugfixes:
 - * Corrected occasional GUI task hangup on STM32F7 targets when compiling with Keil 5.x
 - * Fixed occasional tearing on STM32 F469 EVAL/Discovery boards when using DSI in landscape orientation and single frame buffer mode.
 - * Modified IAR flash loader settings for STM32 F469 boards to enable programming of internal flash (Note: QuadSPI flash must still be programmed from ST-Link Utility as there are no IAR loaders for this)
 - * GPIO class for perf. measurement for STM32F746G-EVAL boards now properly uses the BSP_LED functions. Note that only two signals are active on this board per default because LED2 and LED4 use IO Expander, making them unsuited for measuring performance.
 - \star Removed annoying "Get Alternative File" dialog popups in IAR Workbench when debugging Cortex-M7 applications.

4.4.0

- * Release date: October 6th, 2015
- * New features
 - \star Added support for the Cortex-M7 core.
 - * Introduced concept of "finger size" for touch input. When used,
 TouchGFX will attempt to find touchable widgets in the area
 surrounding the reported x,y coordinates, so users no longer have
 to click precisely on a widget. This feature makes it substantially
 easier to hit small buttons.
 See HAL::setFingerSize().
 - * Supports Visual Studio 2015
 - * Visual Studio projects for Demos and Examples now include Application.props under Resources for quick access. As always a rebuild might be required when altering the contents of Application.props.
 - * Support for Bitmap Fonts in BDF format. If the requested font size is not available in the font file, the font converter will write the supported font size(s) in the error message. See the example monochrome_example for usage.

- * Generating assets now issues better error messages when spaces are detected in paths and file names.
- * All ST boards can now be flashed from the command line provided that ST-Link Utility Release 3.7 has been installed. Simply use 'make -f target/ST/<board>/Makefile flash' to build and flash your application to the connected board. If timeouts occur during flashing, go to Device Manager in Windows and disable "MBED microcontroller USB Device" under "Disk drives".
- * New touchgfx-env version 2.5 available with new gcc cross compiler version 4.9.3. The older version 4.8.4 could generate invalid code for Cortex-M7 cores in rare cases.

* Board support

- * Added support for the STM32F7xx processors
- \star Added support for the STM32F746G-DISCO and STM32756G-EVAL boards
- \star Added support for the STM32F469 processor with DSI displays
- * Added support for the STM32469I-EVAL and STM32469I-Discovery boards

* Bugfixes

- * TextureMapper and ScaleableImage now draws images correctly when using "rotate90".
- * Fixed potential initialization order bug in STM32F4DMA.cpp
- \star Fixed bug that limited number of glyphs in a single font to 32768. Now supports 65536 glyphs per font as intended.
- * Fixed bug that caused hal.lockDMAToFrontPorch(false) to not have any effect in single frame buffer mode.
- * ButtonWithLabel correctly center texts vertically if text contains newlines

4.3.0

* Release date: June 8th, 2015

* New features

* TextureMapper widget added.

The TextureMapper is a highly optimized image renderer that can be used for displaying an image that is scaled and/or rotated in two or three dimensions during run time. This can be used for doing advanced rotation animations of images. See manual or texture_mapper_example for more information. LCD has new methods for drawing triangles and corresponding scan lines, drawTextureMapTriangle and drawTextureMapScanLine

* Alpha Channel Dithering

Images with alpha channel can now get the alpha channel dithered for smoother alpha gradients, see examples or Application Development section in manual for details

- * Compression of 1BPP (monochrome) bitmaps
 Added image format option of BW_RLE, which will cause bitmaps to
 be automatically run-length encoded if that takes up less space than
 the regular per-pixel format. Yields substantially smaller bitmap
 footprint in many cases. See advanced chapter in manual for details.
- * Slider widget added.
- See manual or slider_example for more information.
- * Makefiles has been updated to work with make-4.1.
- \star Added support for the LPC4088 processor and the Embedded Artists LPC4088 Display Module board.
- * Individual bitmaps can now be placed in internal flash instead of external by having the bitmap file name include the string ".int."
- * MoveAnimator, FadeAnimator and ZoomAnimationImage now have a cancelMoveAnimation/cancelFadeAnimation/cancelZoomAnimation method.

* Update procedure

- * Compatible with existing 4.X applications. Just replace the touchgfx folder.
- * Check Known Issues in the documentation.

* Info

 \star The evaluation version of TouchGFX is now distributed with source code for the hardware abstraction layer instead of a precompiled library. This makes it possible to port the evaluation version

to custom hardware instead of it being limited to the supported eval boards only. Instead, the evaluation version now has a TouchGFX watermark which will appear occasionally.

- * Memory consumption reduced due to improved rendering algorithm. Will typically allow GUI task stack to be reduced by around 1400 bytes compared to version 4.2.0 (depending on actual application). Additionally the statically allocated memory is also reduced by around 1KB.
- * Maximum number of visible widgets limit of 150 removed.
- * Added two new demos for 640x480 and 480x272 resolutions showcasing new features, graphs, internationalization and custom widgets.
- * Drawable.setPosition() now calls setXY(), setWidth() and setHeight() for easier subclassing.
- \star AbstractPainterRGB565 and AbstractPainterBW are recommended as base classes when implementing your own painters.
- * CanvasWidgets now have setAlpha() and getAlpha() methods. Your custom Painter classes must implement this, or inherit from the AbstractPainterRGB565 class
- \star Maximum number of registered timer widgets increased from 16 to 24.
- * touchgfx-env updated to 2.4. The environment does not beep anymore.
- \star Board Support Package for STM324x9I-EVAL is now based on the STMCubeF4 drivers.

* Bugfixes

- * Screen::handleGestureEvent now converts x/y to relative coordinates
- \star Fixed bug when drawing several objects on the same canvas using moveTo() more than once.
- * ZoomAnimationImage movement relative to scaling did not use correct easing equation.
- * PainterRGB565 did not blend green alpha correctly.
- * RadioButtonGroup now initializes callbacks to zero.
- * ScalableImage now works with bitmaps with transparancy.
- * AnimatedImage would display the start and end of an animation twice.
- * Default implementation of CanvasWidget::getMinimalRect() returned coordinates relative to its parent, not itself.
- * ScrollableContainer erroneously unregistered itself as a timer widget at every tick, which made it difficult to use with other timer-based operations.

* Performance

* ScalableImage and ZoomAnimationImage has been optimized for better performance.

4.2.0

* Release date: January 14th, 2015

* Performance

 \star Substantially improved rendering performance, which in most cases will result in a 25% reduction of time it takes to render a frame.

NOTE: This optimization does not necessarily work on all targets so it must be manually enabled. See the "Optimization" chapter in the porting guide for how to enable this optimization for existing portings. It is STRONGLY recommended that the optimization is enabled. This optimization is enabled for all appropriate evaluation boards in the 4.2.0 board packages.

* Major new features

- * Added CanvasWidgets for smooth, anti-aliased drawing of geometric shapes. Currently Line, Circle and a more generic Shape have been implemented. CanvasWidgets can be painted with a solid color (+ alpha), a bitmap (including alpha) or a custom painter. Read more on Canvas Widgets and Painters in the documentation.
- * Added support for the Keil compiler and uVision4 IDE. Please refer to the "Supported Hardware" section of the TouchGFX Distribution chapter in the documentation for a list of Keil-supported targets.

* New features

 \star It is now possible to specify an animation start delay on

- ZoomAnimationImage, MoveAnimator and FadeAnimator.
- * Added Board support for 4.3" TouchGFX Demo board w. LPC4350 (No internal flash)
- * RadioButton and RadioButtonGroup widgets added. See app/examples/radio_button_example and documentation.
- * LPC43XX and LPC1788 can now fill rectangles using DMA.
- * Visual Studio 2013 is now supported.
- * Preliminary support for Visual Studio 2015 Preview version.
- * Improved performance when generating assets.
- * New canvas_widget_example added to the example directory.
- * The "using namespace touchgfx" present in various header files can now be avoided by defining the symbol NO_USING_NAMESPACE_TOUCHGFX in your project.

* TouchGFX env

* The message displayed when starting a shell has been fixed with correct path to examples.

* Bugfixes

- \star Fixed bug in simulator for 1bpp displays when width and/or height was not not a multiple of 8.
- * Fixed bug in ScrollableContainer where CANCEL events where not always delegated to correct child, causing e.g. buttons to remain pressed when dragging outside SC area.
- * Fixed bug when rendering chromArt fonts with a rotated display.
- * Fixed bug Keyboard widget setTouchable(false) had no effect.
- * Freescale K70 DMA now checks the appropriate DONE bit in TCDO_CSR.
- \star On ST processors fixed bug with rotated texts rendered by ChromArt when in non-native display orientation.

* Board support

* Embedded Artists LPC4357DevKit board package: CPU clocked to 204Mhz (previously 96Mhz). Now uses SPIFI flash instead of NOR.

* Update procedure

 \star Compatible with existing 4.X applications. Just replace the touchgfx folder.

* Info

* Documentation has been updated.

4.1.1

* Release date: October 29th, 2014

* New features

- * Mixin: MoveAnimator added. The MoveAnimator mixin makes the template class T able to animate a movement from its current position to a specified end position. See app/example/move_fade_example.
- * Mixin: FadeAnimator added. The FadeAnimator mixin makes the template class T able to animate an alpha fade from its current alpha value to a specified end alpha value. See app/example/move_fade_example.
- \star ScalableImage and ZoomAnimationImage now support alpha per pixel bitmaps and alpha per bitmap
- * ScalableImage and ZoomAnimationImage now support ARGB8888 format bitmaps

* Bugfixes

- \star Fixed a bug causing the Keyboard widget to render incorrectly in rare cases.
- * Fixed a bug causing drag event coordinates to be incorrect for widgets when placed in a Container with coords != {0,0} which itself was placed in a ScrollableContainer.
- * The Application class now properly keeps track of number of times registerTimerWidget vs. unregisterTimerWidget is called for a given widget, meaning that if registered several times it now requires same number of unregisters before widget no longer receives tick events.
- * Some ZoomAnimationImage functions were not virtual as they should

be.

- \star Some widgets were missing certain getter functions.
- * Update procedure
 - \star Compatible with existing 4.X applications. Just replace the touchgfx folder.

4.1.0

- * Release date: October 17th, 2014
- * New features
 - * Now supports monochrome 1BPP displays. See manual for details.
 - * Support for dynamic screen orientation change (landscape/portrait)
 - * Support for scaling images (See ScalableImage and ZoomAnimationImage drawables)
- * Demo
 - * Home Control Demo now support 640x480 mode.
 - * Home Control Demo now supports STM324xI-EVAL 5.7" board.
- * Board support changes
 - * Added support for STM324xI-EVAL 5.7" board (IAR+gcc).
 - * Added gcc support for the EmbeddedArtists LPC4357DevKit board.
 - * Optimized SPIFI initialization for TouchGFX eval board.
- * Bugfixes
 - \star Adding a persistent Drawable to a ScrollableContainer could cause assertion
 - * Support for much larger fonts
- * Update procedure
 - \star Compatible with existing 4.X applications. Just replace the touchgfx folder.

4.0.0

- * Release date: September 26th, 2014
- * New features
 - * TouchEvent refactoring (API breaking):
 - * Drawable::setActive is renamed to Drawable::setTouchable
 - * Drawable::isActive is renamed to Drawable::isTouchable
 - * Drawable::hijackTouchEvent is deprecated
 - \star Drawables are now per default not touchables
 - \star TouchEvents are now always propagated to all containers children
 - * Language specific typography and alignment columns support added to text converter. Read more about this feature in the documentation.
 - * Font rendering has been vastly improved with regards to font shapes and kerning.
 - * Simulator assert check on new view/presenter/transition size when doing screen transition. Failed assert checks probably due to missing definition of view/presenter/transition in FrontEndHeap.
 - * TextArea and ButtonWithLabel now support text rotated 0, 90, 180 or 270 degrees.
 - * Text centering on ButtonWithLabel has been improved in special cases.
 - \star Hardware Accelerated text rendering (4 and 8bpp) on supported ST platforms.
 - \star Ability to cache all items in the bitmap database in external RAM.
 - * Support for Freescales K70 MCU.
 - * Translation Sheet: Instances of "\<" and "\>" are converted into "<" and ">" respectively. This enables literal translated strings such as "<Not a wildcard>" using "\<Not a wildcard\>".
 - * Support for NXP LPC18XX series of MCU's.
- * Bugfixes
 - \star Rendering error of images with odd width and alpha value less than

255

- * Correct handling of TextArea::getTextHeight in case of non initialized textArea
- * TextAreaWithWildcard::getTextWidth now includes the width of the wildcard text
- * gcc Makefiles now includes *.BMP and *.PNG from image assets.
- \star Do not trim leading and trailing white space from any translations in the texts sheet.
- * Font converter did not generate font data properly for 8bpp.
- * ButtonWithIcon::setBitmaps Suppress IAR warning for intentional virtual function override.
- * ButtonWithIcon optimized draw functionality
- \star In extremely rare cases text could be written slightly outside the text area
- * Update procedure
 - * Due to the TouchEvent refactoring you have to rename functions accordingly. You also need to state in any custom widget or containers if they need to receive touch events. If you were using hijackTouchEvent to prevent children of getting touch events, you now need to make sure that all children is not touchable instead.
 - \star Main.cpp for simulators need to be updated by replacing the line:

 $\label{thm:continuous} \begin{tabular}{ll} TypedText::registerTypedTextDatabase::getInstance(), TypedTextDatabase::getInstance(), TypedTextD$

Texts::setLanguage(0)

You can also specify a specific language from your text database e.g.

Texts::setLanguage(GB)

In that case you also need to:

#include <texts/TextKeysAndLanguages.hpp>

* Rebuild entire project.

- * Info
 - * The TouchGFX Manual has been updated considerably.

3.1.0

* New features

- * Added support for FDI uEZGUI-1788-70WVT eval board (NXP LPC-1788 Cortex M3).
- * Added support for Mjolner TouchGFX Demo Board Rev. 1.1 eval board (NXP LPC-4353 Cortex M4/M0 4.3").
- * Bugfixes
 - \star Visual Studio build now rebuild BitmapDatabase.h when new images are added to the assets/images folder.
- * Update procedure
 - * Only if using Visual Studio: Update TouchGFXReleasePath in your Visual Studio .props file. Simply edit the file in a text editor. The path should be extended with "touchgfx\". See the template_application for inspiration.
 - * Only if using Visual Studio: Update your Visual Studio project file (.vcxproj file). Simply edit the file in a text editor. Replace all paths on the form
 - $\verb|"$ (TouchGFXReleasePath) \land config\msvs\land touchgfx_prebuild.targets \verb|"with"|$
 - "\$(TouchGFXReleasePath)\config\msvs\touchgfx_prebuild.targets".
- * Info
- * Hardware Abstraction Layer architecture has been reworked so that all common code for various hardware components (MCUs and drivers) is now shared across different target boards. This greatly simplifies the porting effort for new/custom boards as long as they contain one or more hardware components already supported by TouchGFX.

3.0.0

* New features

- * Visual Studio 2010/2012 support.
- * Added support for png images with alpha channel.
- * Added support for subfolders in assets/bitmaps folder
- * Added support for ST STM32F4X9I-EVAL eval board.
- * Added support for Robert Penners Easing Equations (see touchqfx/EasingEquations.hpp).
- * Image converter: Added sanity check of input image file names, must not start with digit and must be alphanumeric.
- * Image converter: Added checking against case insensitively file name duplicates in input list.
- * Text converter: Added build stopping sanity checks for bpp and font size values.
- * ScrollableContainer: Now supports setScrollbarPadding, setScrollbarWidth, setScrollbarColor, and setScrollbarAlpha.
- \star ScrollableContainer: Set default value of ScrollThreshold to 5 pixels, instead of 1.
- * Added support for alpha blending of fonts (TextArea::setAlpha(uint8_t alpha))
- * ImageConvert support two different output formats: RGB565 and ARGB8888
- * ImageConvert two options added to control output format for images with/without an alpha channel
- * Touchgfx environment under MinGW is updated due to linker errors for large projects. g++ version is updated from 4.6.2 -> 4.8.1
- * Internal RAM footprint improvements
- * Structural changes of target library and hardware abstraction layers

* Bugfixes

- * Fill operation (Box widget) resulted in a crash on the lpc4357_emb_artist board
- * Textconvert & fontconvert: Different typographies may now have identical properties.
- * Imageconvert & fontconvert: Better error handling for POSIX compliant platforms
- * HALSDL: Do not overflow key data type.
- * LanguageXX.cpp files now end with a newline (removing warnings).
- * TextArea::draw now handles non initialized TypedText correctly.

* Update procedure

- * The folders assets/bitmaps and generated/bitmaps must be renamed to assets/images and generated/images.
- * Upgrade TouchGFX environment to version 2.0
- * Update any application Makefile to adhere with the Makefile specified in the updated template_application
- * Rebuild entire project
- * Convert bmp images that contains the former transparent color to png images that uses alpha channels. This can be done automatically using a free tool called imagemagick. More info and hints can be acquired by writing touchgfx-support@mjolner.com
- * Custom HAL implementations must be updated to conform with the new structure

* Info

* The "magic" transparent color that was previously used for transparent color in the bmp format is no longer supported. Instead use png images with alpha channel.

2.2.0

* New features

- \star Added support for portrait mode with landscape displays at zero performance/resource cost.
- * Added kerning support.
- * Added Keyboard example (with IAR project for the Energy Micro DK3750 eval board)
- \star Changed interface for blitCopy method in LCD.
- \star Removed SyncBackBuffer method from HAL.
- * Removed clearLCD method from LCD.
- * Removed fillGradientRect method from LCD.
- * ScrollableContainer supports setScrollbarsVisible(bool visible).

2.1.0

 \star First release of TouchGFX as a commercially available framework

Chapter 2

Globals

Member touchgfx::touchgfx_generic_init (DMA_Interface &dma, LCD &display, TouchController &tc, int16← _t width, int16_t height, uint16_t *bitmapCache, uint32_t bitmapCacheSize, uint32_t numberOf← DynamicBitmaps=0)

22 Globals

Chapter 3

Namespace Index

3.1	Names	nace	l ist
U. I	Hailies	pace	∟ıəı

Here is	s a list	t of all	documented	namespaces	with	brief	descript	ions:

t-	uc	ha	ıfv
ιU	uu	ΗU	II A

The global touchgfx namespace.	All TouchGFX framework classes and global functions are	
placed in this namespace		4

24 Namespace Index

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:
AbstractPainter
AbstractPainterABGR2222
PainterABGR2222
PainterABGR2222Bitmap
AbstractPainterARGB2222
PainterARGB2222
PainterARGB2222Bitmap
AbstractPainterARGB8888
PainterARGB8888
PainterARGB8888Bitmap
PainterARGB8888L8Bitmap
AbstractPainterBGRA2222
PainterBGRA2222
PainterBGRA2222Bitmap
AbstractPainterBW
PainterBW
PainterBWBitmap
AbstractPainterGRAY2
PainterGRAY2
PainterGRAY2Bitmap
AbstractPainterGRAY4
PainterGRAY4
PainterGRAY4Bitmap
AbstractPainterRGB565
PainterRGB565
PainterRGB565Bitmap
PainterRGB565L8Bitmap
AbstractPainterRGB888
PainterRGB888
PainterRGB888Bitmap
PainterRGB888L8Bitmap
AbstractPainterRGBA2222
PainterRGBA2222
PainterRGBA2222Bitmap
AbstractPartition
Partition < ListOfTypes, NUMBER_OF_ELEMENTS >

26 Hierarchical Index

AnimationTextureMapper::AnimationSetting14
Bitmap
Bitmap::BitmapData
BlitOp
ButtonController
Buttons
Bitmap::CacheTableEntry
Keyboard::CallbackArea
Canvas
Cell 22 Color 24
colortype
CWRUtil
DebugPrinter
LCD24DebugPrinter
DisplayTransformation
DMA_Interface
NoDMA
DMA_Queue
LockFreeDMA_Queue
Drawable
Container
MoveAnimator< touchgfx::Container >
AbstractButtonContainer
ClickButtonTrigger
RepeatButtonTrigger
ToggleButtonTrigger
TouchButtonTrigger
AbstractClock
AnalogClock
DigitalClock
AbstractProgressIndicator
AbstractDirectionProgress
BoxProgress
ImageProgress
CircleProgress
LineProgress
TextProgress
CacheableContainer
DrawableList
Keyboard
ListLayout
ModalWindow
ScrollableContainer
ScrollBase
ScrollList
ScrollWheelBase
ScrollWheel
ScrollWheelWithSelectionStyle
SlideMenu
Slider
SwipeContainer
ZoomAnimationImage
Widget
AbstractButton
Button

4.1 Class Hierarchy 27

ButtonWithIcon	. 189
ButtonWithLabel	
RepeatButton	
ToggleButton	. 919
RadioButton	
TouchArea	. 924
Box	
BoxWithBorder	
CanvasWidget	
AbstractShape	
Shape < POINTS >	
Circle	
Line	
CanvasWidgetRenderer	
CoverTransition< templateDirection >::FullSolidRect	
Image	
AnimatedImage	
CacheableContainer::CachedImage	
TiledImage	
PixelDataWidget	
ScalableImage	
SnapshotWidget	
TextArea	
TextAreaWithWildcardBase	
TextAreaWithOneWildcard	
TextAreaWithTwoWildcards	
TextureMapper	
AnimationTextureMapper	. 145
DrawableListItemsInterface	317
DrawableListItems < TYPE, SIZE >	. 315
DrawingSurface	318
Bitmap::DynamicBitmapData	319
EasingEquations	319
Edge	335
Event	337
ClickEvent	. 242
DragEvent	. 287
GestureEvent	. 364
FlashDataReader	342
Font	344
ConstFont	. 251
InternalFlashFont	. 426
FontManager	353
FontProvider	354
FrameBufferAllocator	355
ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >	
GenericCallback< T1, T2, T3 >	
Callback< dest_type, T1, T2, T3 >	
GenericCallback< const T &, const touchgfx::ClickEvent &>	
GenericCallback< const touchgfx::AbstractButton & >	
GenericCallback< const touchgfx::AbstractButton & void, void >	
Callback< touchgfx::RadioButtonGroup, const touchgfx::AbstractButton & >	. 201
GenericCallback< const touchgfx::AbstractButtonContainer & >	
GenericCallback< const touchgfx::AbstractButtonContainer & >	

28 Hierarchical Index

$\label{lem:constraint} Generic Callback < const touchgfx:: Animation Texture Mapper \& > \ \dots \$	358
GenericCallback< const touchgfx::DragEvent &>	358
$\label{eq:GenericCallback} Generic Callback < const touchgfx:: Fade Animator < T > \& > $	358
$\label{eq:const_touchgfx::MoveAnimator} Generic Callback < const touchgfx:: MoveAnimator < T > \& > \ \dots \dots$	358
$\label{lem:constrainer} Generic Callback < const touchgfx:: Move Animator < touchgfx:: Container > \& > . \ . \ . \ . \ . \ . \ . \ . \ . \ .$	358
$\label{lem:constrainer} Generic Callback < const touchgfx:: Move Animator < touchgfx:: Container > \&, void, void > \ . \ . \ . \ . \ . \ . \ . \ . \ . \$	358
$lem:const_const$. 201
GenericCallback< const touchgfx::SlideMenu & >	358
GenericCallback< const touchgfx::Slider &, int >	
GenericCallback< const touchgfx::ZoomAnimationImage &>	358
GenericCallback< int16_t >	
GenericCallback < T1 >	358
Callback< dest_type, T1, void, void >	. 204
GenericCallback< T1, T2 >	
Callback< dest_type, T1, T2, void >	. 203
GenericCallback< T1, T2, void >	
GenericCallback< T1, void, void >	
GenericCallback< touchgfx::Drawable &, void, void >	
Callback< touchgfx::CoverTransition, touchgfx::Drawable &>	
Callback< touchgfx::SlideTransition, touchgfx::Drawable &>	
$\label{lem:control_def} Generic Callback < touchgfx::Drawable List I tems Interface *, int 16_t, int 16_t > \dots $	
GenericCallback< Unicode::UnicodeChar >	
GenericCallback< void >	
GenericCallback<>	
Callback< dest_type, void, void >	. 206
Gestures	
GlyphNode	368
GPIO	371
Gradients	373
HAL	374
HALSDL2	. 402
I2C	411
Scanline::iterator	
JSMOCHelper	
KerningNode	
Keyboard::Key	
Keyboard::KeyMapping	
Keyboard::KeyMappingList	
Keyboard::Layout	440
LCD	441
LCD16bpp	. 457
LCD16bppSerialFlash	
LCD1bpp	
LCD24bpp	
LCD2bpp	
LCD32bpp	
LCD4bpp	
LCD8bpp ABGR2222	. 541
LCD8bpp ARGB2222	. 552
LCD8bpp_BGRA2222	
LCD8bpp_RGBA2222	
LED	
Matrix4x4	
MCUInstrumentation	
MVPHeap	
OSWrappers	

4.1 Class Hierarchy 29

Outline	636
$Pair < T1, T2 > \dots $	
$Pair < int16_t, int16_t > \dots $	
Point	
Point3D	
Presenter	
CWRUtil::Q10	
CWRUtil::Q15	
CWRUtil::Q5	
Quadruple	
Point4	
Vector4	
RadioButtonGroup < CAPACITY >	
Rasterizer	
Rect	
Renderer	
Scanline	
Screen	_
View< T >	
$AbstractShape::ShapePoint < T > \dots \dots$	
LCD::StringVisuals	
TextProvider	
Texts	
TextureSurface	
TouchCalibration	
TouchController	
I2CTouchController	413
NoTouchController	631
SDL2TouchController	828
SDLTouchController	829
Transition	930
CoverTransition < templateDirection >	260
NoTransition	
SlideTransition< templateDirection >	856
TypedText	
TypedText::TypedTextData	
UIEventListener	
Application	
MVPApplication	
• •	
Unicode	
Vector< T, capacity >	
Vector< touchgfx::Drawable *, MAX_TIMER_WIDGETS >	
Vector< touchgfx::Rect, 8 >	954
T Andreaded the care Button Ot do a T S	4.40
AnimatedImageButtonStyle < T >	
BoxWithBorderButtonStyle < T >	
ClickListener < T >	
Draggable < T >	
Snapper< T >	
FadeAnimator< T >	
IconButtonStyle < T >	
ImageButtonStyle < T >	
MoveAnimator< T >	
PreRenderable < T >	
TextButtonStyle < T >	
TiledImageButtonStyle < T >	917

TwoWildcardTextButtonStyle $<$ T $>$. 9	32
WildcardTextButtonStyle< T >														9	62

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AbstractButton	
This class defines an abstract interface for button-like elements	69
AbstractButtonContainer	
An abstract button container	71
AbstractClock	
Superclass of clock widgets	73
AbstractDirectionProgress	
An abstract direction progress	76
AbstractPainter	
An abstract class for creating painter classes for drawing canvas widgets	77
AbstractPainterABGR2222	
A Painter that will paint using a color and an alpha value	80
AbstractPainterARGB2222	
A Painter that will paint using a color and an alpha value	83
AbstractPainterARGB8888	
A Painter that will paint using a color and an alpha value	86
AbstractPainterBGRA2222	
A Painter that will paint using a color and an alpha value	89
AbstractPainterBW	
A Painter that will paint using a color on a LCD1bpp display	92
AbstractPainterGRAY2	
A Painter that will paint using a color and an alpha value	94
AbstractPainterGRAY4	
A Painter that will paint using a color and an alpha value	96
AbstractPainterRGB565	
A Painter that will paint using a color and an alpha value	99
AbstractPainterRGB888	
A Painter that will paint using a color and an alpha value	102
AbstractPainterRGBA2222	
A Painter that will paint using a color and an alpha value	104
AbstractPartition	
This type defines an abstract interface to a storage partition for allocating memory slots of equal	
size	108
AbstractProgressIndicator	
An abstract progress indicator	114
AbstractShape Simple widget capable of drawing a abstractShape	120
auricie wiodel capable of drawing a apstractablane	120

32 Class Index

AnalogClock	
An analog clock	130
AnimatedImage	400
A widget capable of basic animation using a range of bitmaps	138
An animated image button style. An animated image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show the first or last image of an animated image depending on the state of the button (pressed or released). When the state changes the button will show the sequence of images in forward or	
reversed order	143
AnimationTextureMapper::AnimationSetting Information about how a specific animation parameter should be animated	144
AnimationTextureMapper	1 45
A texture mapper with animation capabilities	145
Main interface for manipulating screen contents	149
Bitmap This class provides a proxy object for a bitmap image	157
Bitmap::BitmapData	107
Data of a bitmap	169
BlitOp	
BlitOp instances carry the required information for performing operations on the LCD (frame buffer) using DMA	170
Box	
Simple widget capable of showing a rectangle of a specific color and an optional alpha	171
BoxProgress A box progress	175
BoxWithBorder	170
A box with border	178
A box with border button style	182
Button	
A button with two states	184
ButtonController	4.07
Interface for sampling external key events	187
A buttons	189
ButtonWithIcon	
A Button specialization that also displays an icon on top of the button bitmap	189
ButtonWithLabel	100
A Button specialization that also displays a text on top of the button bitmap	193
A CacheableContainer is a Container that can have its drawing redirected into a Bitmap	196
CacheableContainer::CachedImage	
A CachedImage is a specialized Image object that exposes the setupDrawChain() method	199
Bitmap::CacheTableEntry	
Cache bookkeeping	200
A Callback is basically a wrapper of a pointer-to-member-function	201
Callback< dest_type, T1, T2, void >	
A Callback is basically a wrapper of a pointer-to-member-function	203
Callback< dest_type, T1, void, void >	
A Callback is basically a wrapper of a pointer-to-member-function	204
Callback< dest_type, void, void > A Callback is basically a wrapper of a pointer-to-member-function	206
Keyboard::CallbackArea	
Mapping from rectangle to a callback method to execute	208

5.1 Class List

Canvas		
	Class for easy rendering using CanvasWidgetRenderer	208
CanvasV	Widget Class for drawing complex polygons on the LCD using CanvasWidgetRenderer	211
CanvasV	VidgetRenderer Class for supporting drawing of figures	215
Cell	A pixel cell	220
Circle	Simple widget capable of drawing a circle	222
CirclePro		235
ClickBut	tonTrigger A click button trigger	242
ClickEve	ent	242
ClickList	ener< T >	
Color	Mix-in class that extends a class with a click action event	246
colortype	Contains functionality for color conversion	247
ConstFo	Type for representing a color	249
Constru	A ConstFont is a Font implementation that has its contents defined at compile-time and usually placed in read-only memory	251
Containe	er A Container is a Drawable that can have child nodes	254
CoverTra	ansition < templateDirection > A Transition that slides from one screen to the next	260
CWRUtil		
DebugPı		262
DigitalCl	The class DebugPrinter defines the interface for printing debug messages on top of the frame-buffer	270
	A digital clock	272
	Transformation Defines transformations from display space to frame buffer space	277
DMA_Int	DMA_Interface provides basic functionality and structure for processing "blit" operations using DMA	279
D.WQ.	This class provides an interface for a FIFO (circular) list used by DMA_Interface and descendants for storing BlitOp's	285
DragEve	ent A drag event	287
Draggab	ole < T > Mix-in class that extends a class to become draggable	291
Drawable	e	
Drawable		
Drawable	A container able to display many items using only a few drawables eListItems < TYPE, SIZE >	306
Drawable	An array of drawables used by DrawableList	315
	A drawable list items interface	317

34 Class Index

Drawing:	Surface	
	The destination of a draw operation. Contains a pointer to where to draw and the stride of the drawing surface	318
Bitmap::	DynamicBitmapData Data of a dynamic bitmap	319
EasingE	·	
_	Defines the "Penner easing functions", which are a de facto standard computing aesthetically pleasing motion animations	319
Edge		
	An edge contains information about one edge, between two points, of a triangle, as well as information about how to interpolate values when moving in the vertical direction	335
Event		
FadeAni	Simple base class for events	337
ElachDa	A FadeAnimator makes the template class T able to animate an alpha fade taReader	339
riasiiDa		342
Font	This class is an abstract interface for a class reading data from a flash	342
	The font base class	344
FontMan	nager	
FontProv	This class is the entry point for looking up a font based on a font id	353
1 01111 101	A generic pure virtual definition of a FontProvider	354
FrameRi	ufferAllocator	004
	This class is an abstract interface for a class allocating partial framebuffer blocks	355
Coverira	ansition< templateDirection >::FullSolidRect	
	A Widget that returns a solid rect of the same size as the application	357
Generic(Callback < T1, T2, T3 > GenericCallback is the base class for callbacks	358
Generic(Callback< T1, T2, void >	
Generic(GenericCallback is the base class for callbacks	359
	GenericCallback is the base class for callbacks	361
Generic(Callback< void > GenericCallback is the base class for callbacks	362
Gesturel		302
Gesturei	A gesture event	364
Gestures		304
	This class implements the detection of gestures	366
GlyphNo		
GPIO	Struct providing information about a glyph	368
	Interface class for manipulating GPIOs in order to do performance measurements on target	371
Gradient	ds and the same of	
	Gradients contains all the data to interpolate u,v texture coordinates and z coordinates across a planar surface	373
HAL	planar curiace	0,0
	Hardware Abstraction Layer	374
HALSDL	•	•
TH LODE	HAL implementation for the TouchGFX simulator	402
I2C		
IOCT :	Platform independent interface for I2C drivers	411
12C louc	hController	445
loon Dust	Specific I2C-enabled type of Touch Controller	413
iconButte	onStyle< T > An icon button style	415
Image	Simple widget capable of showing a bitmap	<u>4</u> 17
	omple waget capable of showing a bitmap	717

5.1 Class List 35

ImageButtonStyle< T >	
An image button style	420
ImageProgress	
An image progress	422
InternalFlashFont	
An InternalFlashFont has both glyph table and glyph data placed in a flash which supports random access read (i.e. not a NAND flash)	426
Scanline::iterator	
An iterator to help go through all the elements that make up a Scanline	428
JSMOCHelper	
Helper class providing caching of certain information while the JSMOC algorithm runs during draw operations	429
KerningNode	
Structure providing information about a kerning for a given char pair	432
Keyboard::Key	
Mapping from rectangle to key id	432
Keyboard The back and assisted to the state of the state	400
The keyboard provides text input for touch devices	433
Keyboard::KeyMapping	4.40
Mapping from key id to Unicode character	440
Keyboard::KeyMappingList	440
List of KeyMappings to use	440
Keyboard::Layout Definition of the keyboard layout. The keyboard can handle changing layoute as different key.	
Definition of the keyboard layout. The keyboard can handle changing layouts, so different keyboard modes can be implemented by changing layouts and key mappings	440
LCD	440
This class contains the various low-level drawing routines for drawing bitmaps, texts and rectan-	
gles	441
LCD16bpp	
This class contains the various low-level drawing routines for drawing bitmaps	457
LCD16bppSerialFlash	
This class contains the various low-level drawing routines for drawing bitmaps	470
LCD1bpp	
This class contains the various low-level drawing routines for drawing bitmaps, texts and rectan-	
gles	482
LCD24bpp	
This class contains the various low-level drawing routines for drawing bitmaps	493
LCD24DebugPrinter	
The class LCD24DebugPrinter implements the DebugPrinter interface for printing debug messages on top of 24bit framebuffer	505
LCD2bpp	
This class contains the various low-level drawing routines for drawing bitmaps	506
LCD32bpp	
This class contains the various low-level drawing routines for drawing bitmaps	517
LCD4bpp	
This class contains the various low-level drawing routines for drawing bitmaps	530
LCD8bpp_ABGR2222	
This class contains the various low-level drawing routines for drawing bitmaps	541
LCD8bpp_ARGB2222	
This class contains the various low-level drawing routines for drawing bitmaps	552
LCD8bpp_BGRA2222	
This class contains the various low-level drawing routines for drawing bitmaps	563
LCD8bpp_RGBA2222 This place contains the various low level drawing routines for drawing hitmans	E7.
This class contains the various low-level drawing routines for drawing bitmaps	574
A led	585
Line	500
Simple CanvasWidget capable of drawing a line	500
Simple Calivas widget capable of drawing a line	500

36 Class Index

LineProgress	
A line progress	598
ListLayout	
This class provides a layout mechanism for arranging Drawable instances adjacently in the specified Direction	604
LockFreeDMA_Queue	
This implements a simple lock-free FIFO queue (single producer, single consumer)	607
This class is partial framebuffer allocator using multiple blocks	609
Matrix4x4	
This class represents row major 4x4 homogeneous matrices	611
MCUInstrumentation Interface for instrumenting processors to measure MCU load via measured CPU cycles	616
ModalWindow	010
Container for displaying a modal window and hijacking touch event to underlaying view and	
widgets	618
MoveAnimator < T > A MoveAnimator makes the template class T able to animate a movement	623
MVPApplication	023
A specialization of the TouchGFX Application class	627
MVPHeap	
Generic heap class for MVP applications	628
This is an "empty" DMA subclass that does nothing except assert if accidentally used	629
NoTouchController	631
NoTransition	
The most simple Transition without any visual effects	633
OSWrappers This class specifies OS wrappers for dealing with the frame buffer semaphore and the VSYNC	
signal	634
Outline	
An internal class that implements the main rasterization algorithm	636
PainterABGR2222 A Painter that will paint using a color and an alpha value	639
PainterABGR2222Bitmap	000
A Painter that will paint using a bitmap	642
PainterARGB2222	
A Painter that will paint using a color and an alpha value	645
A Painter that will paint using a bitmap	648
PainterARGB8888	0.0
A Painter that will paint using a color and an alpha value	652
PainterARGB888Bitmap	055
A Painter that will paint using a bitmap	655
A Painter that will paint using a bitmap	658
PainterBGRA2222	
A Painter that will paint using a color and an alpha value	661
PainterBGRA2222Bitmap A Painter that will paint using a bitmap	665
PainterBW	005
A Painter that will paint using a color on a LCD1bpp display	668
PainterBWBitmap	
A Painter that will paint using a bitmap	670
PainterGRAY2 A Painter that will paint using a color and an alpha value	673
PainterGRAY2Bitmap	5,0
A Painter that will paint using a bitmap	676

5.1 Class List 37

PainterGRAY4	
	679
PainterGRAY4Bitmap	000
·	682
PainterRGB565 A Painter that will paint using a color and an alpha value	686
Painter RGB565Bitmap	000
·	689
PainterRGB565L8Bitmap	
A Painter that will paint using a bitmap	692
PainterRGB888	
·	696
PainterRGB888Bitmap	700
	700
PainterRGB888L8Bitmap A Painter that will paint using a bitmap	703
PainterRGBA2222	700
	707
PainterRGBA2222Bitmap	
A Painter that will paint using a bitmap	710
Pair< T1, T2 >	
	713
Partition < ListOfTypes, NUMBER_OF_ELEMENTS >	
This type provides a concrete Partition of memory-slots capable of holding any of the specified	745
list of types	715
	718
Point	, , ,
	720
Point3D	
·	721
Point4	70
This class represents a homogeneous 3D point	721
	722
Presenter	1 22
The Presenter base class that all application-specific presenters should derive from	724
CWRUtil::Q10	
Defines a number with 10 bits reserved for fraction	725
CWRUtil::Q15	
Defines a number with 15 bits reserved for fraction	728
CWRUtil::Q5	
Defines a number with 5 bits reserved for fraction	/30
Quadruple Base class for homogeneous vectors and points	725
RadioButton	750
	738
RadioButtonGroup < CAPACITY >	
Class for handling a collection of RadioButtons	743
Rasterizer	
1,31	749
Rect	
, , ,	753
Renderer This class template is used basically for rendering scan lines	758
RenderingBuffer	1 30
Rendering buffer wrapper	760

38 Class Index

RepeatButton	
A button with two states	764
RepeatButtonTrigger	
A repeat button trigger	766
ScalableImage	
Widget for representing a scaled version of a bitmap	768
Scanline This place is used to transfer data from place Outline (or a similar one) to the randoming buffer.	770
This class is used to transfer data from class Outline (or a similar one) to the rendering buffer .	773
Screen A Screen represents a full-screen drawable area. Applications create specific screens by sub-	
classing this class	777
ScrollableContainer	,,,
A ScrollableContainer is a container that allows its contents to be scrolled	783
ScrollBase	, 00
A scroll base class	794
ScrollList	
A scrolling menu of drawables	809
ScrollWheel	
A scroll wheel	814
ScrollWheelBase	
A scroll wheel base class	816
ScrollWheelWithSelectionStyle	
A scroll wheel with selection style	820
SDL2TouchController	
TouchController for the simulator	828
SDLTouchController	
TouchController for the simulator	829
Shape < POINTS >	
Simple widget capable of drawing a shape	830
AbstractShape::ShapePoint< T >	000
Defines an alias representing the array of points making up the abstract shape	833
SingleBlockAllocator< block_size, bytes_pr_pixel > This class is partial framebuffer allocator using just one block	834
SlideMenu	034
SlideMenu is a container that has the functionality of being either collapsed or expanded	836
Slider	000
A slider is a graphical element with which the user may set a value by moving an indicator or by	
clicking the slider	846
SlideTransition < templateDirection >	
A Transition that slides from one screen to the next	856
Snapper< T >	
A mix-in that will make class T draggable and able to snap to a position	858
SnapshotWidget	
A widget that is able to make a snapshot of the area the SnapshotWidget covers	861
LCD::StringVisuals	
The visual elements when writing a string	864
SwipeContainer	
A swipe container	865
TextArea	070
This widget is capable of showing a text area on the screen	870
TextArea With One Wildcard TextArea with one wildcard	070
TextArea With Two Wildcards TextArea With Two Wildcards	0/8
TextArea with two wildcards	880
TextArea With Wildcards TextArea With Wildcards TextArea With Wildcard Base	000
Base class for TextAreas displaying texts with one or more wildcards	883
TextButtonStyle < T >	550
A text button style	885
•	

5.1 Class List

TextProgress	
A text progress	888
TextProvider	
The TextProvider is used in drawing basic and wildcard strings	892
Texts	
Class for setting language and getting texts	895
TextureMapper	
Widget capable of drawing a transformed image	897
TextureSurface	
A texture source. Contains a pointer to the data and the width and height of the texture. The alpha channel is used in 565 rendering with alpha. The stride is the width used when moving to	040
the next line of the texture	912
TiledImage Simple widget capable of showing a tiled bitmap	913
TiledImageButtonStyle < T >	910
A tiled image button style	917
ToggleButton	517
A ToggleButton is a Button specialization that swaps the two bitmaps when clicked	919
ToggleButtonTrigger	0.0
A toggle button trigger	922
TouchArea	
Invisible widget used to capture touch events	924
TouchButtonTrigger	
A touch button trigger	927
TouchCalibration	
Calibrates a touch coordinate	928
TouchController	
Basic Touch Controller interface	928
Transition	
Base class for Transitions	930
TwoWildcardTextButtonStyle < T >	
A wildcard text button style	932
TypedText	001
TypedText represents text (as in characters) and typography (as in font and alignment) TypedText::TypedTextData	935
The data structure for typed texts	939
UlEventListener	308
This class declares a handler interface for user interface events	939
Unicode	000
This class provides simple helper functions for working with 16-bit strings	941
Vector < T, capacity >	
A very simple container class using pre-allocated memory	954
Vector4	
This class represents a homogeneous 3D vector	958
View< T >	
This is a generic touchgfx::Screen specialization for normal applications	959
Widget	
A Widget is a Drawable leaf (i.e. not a container)	960
WildcardTextButtonStyle < T >	
A wildcard text button style	962
ZoomAnimationImage	_
Class for optimizing and wrapping move and zoom operations on ScalableImages	965

40 Class Index

Chapter 6

Namespace Documentation

6.1 touchgfx Namespace Reference

The global touchgfx namespace. All TouchGFX framework classes and global functions are placed in this namespace.

Classes

· class AbstractButton

This class defines an abstract interface for button-like elements.

· class AbstractButtonContainer

An abstract button container.

class AbstractClock

Superclass of clock widgets.

class AbstractDirectionProgress

An abstract direction progress.

· class AbstractPainter

An abstract class for creating painter classes for drawing canvas widgets.

• class AbstractPainterABGR2222

A Painter that will paint using a color and an alpha value.

class AbstractPainterARGB2222

A Painter that will paint using a color and an alpha value.

class AbstractPainterARGB8888

A Painter that will paint using a color and an alpha value.

class AbstractPainterBGRA2222

A Painter that will paint using a color and an alpha value.

class AbstractPainterBW

A Painter that will paint using a color on a LCD1bpp display.

• class AbstractPainterGRAY2

A Painter that will paint using a color and an alpha value.

· class AbstractPainterGRAY4

A Painter that will paint using a color and an alpha value.

• class AbstractPainterRGB565

A Painter that will paint using a color and an alpha value.

class AbstractPainterRGB888

A Painter that will paint using a color and an alpha value.

· class AbstractPainterRGBA2222

A Painter that will paint using a color and an alpha value.

· class AbstractPartition

This type defines an abstract interface to a storage partition for allocating memory slots of equal size.

· class AbstractProgressIndicator

An abstract progress indicator.

· class AbstractShape

Simple widget capable of drawing a abstractShape.

class AnalogClock

An analog clock.

· class AnimatedImage

A widget capable of basic animation using a range of bitmaps.

· class AnimatedImageButtonStyle

An animated image button style. An animated image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show the first or last image of an animated image depending on the state of the button (pressed or released). When the state changes the button will show the sequence of images in forward or reversed order.

class AnimationTextureMapper

A texture mapper with animation capabilities.

class Application

The Application class is the main interface for manipulating screen contents.

class Bitmap

This class provides a proxy object for a bitmap image.

struct BlitOp

BlitOp instances carry the required information for performing operations on the LCD (frame buffer) using DMA.

· class Box

Simple widget capable of showing a rectangle of a specific color and an optional alpha.

class BoxProgress

A box progress.

class BoxWithBorder

A box with border.

class BoxWithBorderButtonStyle

A box with border button style.

class Button

A button with two states.

· class ButtonController

Interface for sampling external key events.

class Buttons

A buttons.

· class ButtonWithIcon

A Button specialization that also displays an icon on top of the button bitmap.

class ButtonWithLabel

A Button specialization that also displays a text on top of the button bitmap.

class CacheableContainer

A CacheableContainer is a Container that can have its drawing redirected into a Bitmap.

struct Callback

A Callback is basically a wrapper of a pointer-to-member-function.

struct Callback< dest_type, T1, T2, void >

A Callback is basically a wrapper of a pointer-to-member-function.

struct Callback< dest_type, T1, void, void >

A Callback is basically a wrapper of a pointer-to-member-function.

struct Callback< dest_type, void, void, void >

A Callback is basically a wrapper of a pointer-to-member-function.

class Canvas

Class for easy rendering using CanvasWidgetRenderer.

· class CanvasWidget

Class for drawing complex polygons on the LCD using CanvasWidgetRenderer.

· class CanvasWidgetRenderer

Class for supporting drawing of figures.

struct Cell

A pixel cell.

· class Circle

Simple widget capable of drawing a circle.

class CircleProgress

A circle progress.

class ClickButtonTrigger

A click button trigger.

class ClickEvent

A click event.

· class ClickListener

Mix-in class that extends a class with a click action event.

· class Color

Contains functionality for color conversion.

struct colortype

Type for representing a color.

class ConstFont

A ConstFont is a Font implementation that has its contents defined at compile-time and usually placed in read-only memory.

· class Container

A Container is a Drawable that can have child nodes.

class CoverTransition

A Transition that slides from one screen to the next.

struct CWRUtil

Helper classes and functions for CanvasWidget.

class DebugPrinter

The class DebugPrinter defines the interface for printing debug messages on top of the framebuffer.

· class DigitalClock

A digital clock.

• class DisplayTransformation

Defines transformations from display space to frame buffer space.

class DMA_Interface

DMA_Interface provides basic functionality and structure for processing "blit" operations using DMA.

class DMA Queue

This class provides an interface for a FIFO (circular) list used by DMA_Interface and descendants for storing BlitOp's.

class DragEvent

A drag event.

· class Draggable

Mix-in class that extends a class to become draggable.

• class Drawable

The Drawable class is an abstract definition of something that can be drawn.

· class DrawableList

A container able to display many items using only a few drawables.

· class DrawableListItems

An array of drawables used by DrawableList.

· class DrawableListItemsInterface

A drawable list items interface.

• struct DrawingSurface

The destination of a draw operation. Contains a pointer to where to draw and the stride of the drawing surface.

class EasingEquations

Defines the "Penner easing functions", which are a de facto standard computing aesthetically pleasing motion animations.

struct Edge

An edge contains information about one edge, between two points, of a triangle, as well as information about how to interpolate values when moving in the vertical direction.

· class Event

Simple base class for events.

class FadeAnimator

A FadeAnimator makes the template class T able to animate an alpha fade.

· class FlashDataReader

This class is an abstract interface for a class reading data from a flash.

· class Font

The font base class.

class FontManager

This class is the entry point for looking up a font based on a font id.

class FontProvider

A generic pure virtual definition of a FontProvider.

· class FrameBufferAllocator

This class is an abstract interface for a class allocating partial framebuffer blocks.

· class GenericCallback

GenericCallback is the base class for callbacks.

class GenericCallback< T1, T2, void >

GenericCallback is the base class for callbacks.

class GenericCallback< T1, void, void >

GenericCallback is the base class for callbacks.

class GenericCallback< void >

GenericCallback is the base class for callbacks.

class GestureEvent

A gesture event.

· class Gestures

This class implements the detection of gestures.

struct GlyphNode

struct providing information about a glyph.

· class GPIO

Interface class for manipulating GPIOs in order to do performance measurements on target.

struct Gradients

Gradients contains all the data to interpolate u,v texture coordinates and z coordinates across a planar surface.

· class HAL

Hardware Abstraction Layer.

class HALSDL2

 ${\it HAL}$ implementation for the TouchGFX simulator.

class I2C

Platform independent interface for I2C drivers.

• class I2CTouchController

Specific I2C-enabled type of Touch Controller.

· class IconButtonStyle

An icon button style.

class Image

Simple widget capable of showing a bitmap.

class ImageButtonStyle

An image button style.

class ImageProgress

An image progress.

· class InternalFlashFont

An InternalFlashFont has both glyph table and glyph data placed in a flash which supports random access read (i.e. not a NAND flash).

class JSMOCHelper

Helper class providing caching of certain information while the JSMOC algorithm runs during draw operations.

struct KerningNode

Structure providing information about a kerning for a given char pair.

· class Keyboard

The keyboard provides text input for touch devices.

class LCD

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles.

class LCD16bpp

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD16bppSerialFlash

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD1bpp

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles.

class LCD24bpp

This class contains the various low-level drawing routines for drawing bitmaps.

· class LCD24DebugPrinter

The class LCD24DebugPrinter implements the DebugPrinter interface for printing debug messages on top of 24bit framebuffer.

class LCD2bpp

This class contains the various low-level drawing routines for drawing bitmaps.

· class LCD32bpp

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD4bpp

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD8bpp_ABGR2222

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD8bpp_ARGB2222

This class contains the various low-level drawing routines for drawing bitmaps.

• class LCD8bpp_BGRA2222

This class contains the various low-level drawing routines for drawing bitmaps.

class LCD8bpp_RGBA2222

This class contains the various low-level drawing routines for drawing bitmaps.

· class LED

A led.

class Line

Simple CanvasWidget capable of drawing a line.

class LineProgress

A line progress.

class ListLayout

This class provides a layout mechanism for arranging Drawable instances adjacently in the specified Direction.

class LockFreeDMA_Queue

This implements a simple lock-free FIFO queue (single producer, single consumer).

class ManyBlockAllocator

This class is partial framebuffer allocator using multiple blocks.

· class Matrix4x4

This class represents row major 4x4 homogeneous matrices.

· class MCUInstrumentation

Interface for instrumenting processors to measure MCU load via measured CPU cycles.

class ModalWindow

Container for displaying a modal window and hijacking touch event to underlaying view and widgets.

· class MoveAnimator

A MoveAnimator makes the template class T able to animate a movement.

class MVPApplication

A specialization of the TouchGFX Application class.

class MVPHeap

Generic heap class for MVP applications.

class NoDMA

This is an "empty" DMA subclass that does nothing except assert if accidentally used.

- class NoTouchController
- class NoTransition

The most simple Transition without any visual effects.

class OSWrappers

This class specifies OS wrappers for dealing with the frame buffer semaphore and the VSYNC signal.

class Outline

An internal class that implements the main rasterization algorithm.

class PainterABGR2222

A Painter that will paint using a color and an alpha value.

· class PainterABGR2222Bitmap

A Painter that will paint using a bitmap.

• class PainterARGB2222

A Painter that will paint using a color and an alpha value.

class PainterARGB2222Bitmap

A Painter that will paint using a bitmap.

class PainterARGB8888

A Painter that will paint using a color and an alpha value.

• class PainterARGB8888Bitmap

A Painter that will paint using a bitmap.

class PainterARGB8888L8Bitmap

A Painter that will paint using a bitmap.

• class PainterBGRA2222

A Painter that will paint using a color and an alpha value.

class PainterBGRA2222Bitmap

A Painter that will paint using a bitmap.

class PainterBW

A Painter that will paint using a color on a LCD1bpp display.

class PainterBWBitmap

A Painter that will paint using a bitmap.

class PainterGRAY2

A Painter that will paint using a color and an alpha value.

· class PainterGRAY2Bitmap

A Painter that will paint using a bitmap.

class PainterGRAY4

A Painter that will paint using a color and an alpha value.

· class PainterGRAY4Bitmap

A Painter that will paint using a bitmap.

class PainterRGB565

A Painter that will paint using a color and an alpha value.

class PainterRGB565Bitmap

A Painter that will paint using a bitmap.

class PainterRGB565L8Bitmap

A Painter that will paint using a bitmap.

• class PainterRGB888

A Painter that will paint using a color and an alpha value.

· class PainterRGB888Bitmap

A Painter that will paint using a bitmap.

class PainterRGB888L8Bitmap

A Painter that will paint using a bitmap.

• class PainterRGBA2222

A Painter that will paint using a color and an alpha value.

class PainterRGBA2222Bitmap

A Painter that will paint using a bitmap.

· struct Pair

A simple struct for holding pairs of data.

class Partition

This type provides a concrete Partition of memory-slots capable of holding any of the specified list of types.

· class PixelDataWidget

A widget for displaying a buffer of pixel data.

struct Point

A simple struct containing coordinates.

struct Point3D

A 3D point.

class Point4

This class represents a homogeneous 3D point.

class PreRenderable

This mixin can be used on any Drawable.

class Presenter

The Presenter base class that all application-specific presenters should derive from.

class Quadruple

Base class for homogeneous vectors and points.

class RadioButton

Radio button with two states.

· class RadioButtonGroup

Class for handling a collection of RadioButtons.

class Rasterizer

Polygon Rasterizer that is used to render filled polygons with high-quality Anti- Aliasing.

· class Rect

Class representing a Rectangle with a few convenient methods.

class Renderer

This class template is used basically for rendering scan lines.

class RenderingBuffer

Rendering buffer wrapper.

· class RepeatButton

A button with two states.

· class RepeatButtonTrigger

A repeat button trigger.

class ScalableImage

Widget for representing a scaled version of a bitmap.

· class Scanline

This class is used to transfer data from class Outline (or a similar one) to the rendering buffer.

class Screen

A Screen represents a full-screen drawable area. Applications create specific screens by subclassing this class.

· class ScrollableContainer

A ScrollableContainer is a container that allows its contents to be scrolled.

class ScrollBase

A scroll base class.

· class ScrollList

A scrolling menu of drawables.

· class ScrollWheel

A scroll wheel.

· class ScrollWheelBase

A scroll wheel base class.

• class ScrollWheelWithSelectionStyle

A scroll wheel with selection style.

class SDL2TouchController

TouchController for the simulator.

· class SDLTouchController

TouchController for the simulator.

class Shape

Simple widget capable of drawing a shape.

class SingleBlockAllocator

This class is partial framebuffer allocator using just one block.

• class SlideMenu

SlideMenu is a container that has the functionality of being either collapsed or expanded.

class Slider

A slider is a graphical element with which the user may set a value by moving an indicator or by clicking the slider.

• class SlideTransition

A Transition that slides from one screen to the next.

class Snapper

A mix-in that will make class T draggable and able to snap to a position.

· class SnapshotWidget

A widget that is able to make a snapshot of the area the SnapshotWidget covers.

class SwipeContainer

A swipe container.

· class TextArea

This widget is capable of showing a text area on the screen.

class TextAreaWithOneWildcard

TextArea with one wildcard.

class TextAreaWithTwoWildcards

TextArea with two wildcards.

class TextAreaWithWildcardBase

Base class for TextAreas displaying texts with one or more wildcards.

• class TextButtonStyle

A text button style.

class TextProgress

A text progress.

class TextProvider

The TextProvider is used in drawing basic and wildcard strings.

· class Texts

Class for setting language and getting texts.

class TextureMapper

The TextureMapper class is a widget capable of drawing a transformed image.

struct TextureSurface

A texture source. Contains a pointer to the data and the width and height of the texture. The alpha channel is used in 565 rendering with alpha. The stride is the width used when moving to the next line of the texture.

class TiledImage

Simple widget capable of showing a tiled bitmap.

class TiledImageButtonStyle

A tiled image button style.

· class ToggleButton

A ToggleButton is a Button specialization that swaps the two bitmaps when clicked.

· class ToggleButtonTrigger

A toggle button trigger.

· class TouchArea

Invisible widget used to capture touch events.

class TouchButtonTrigger

A touch button trigger.

· class TouchCalibration

Calibrates a touch coordinate.

· class TouchController

Basic Touch Controller interface.

class Transition

The Transition class is the base class for Transitions.

class TwoWildcardTextButtonStyle

A wildcard text button style.

class TypedText

TypedText represents text (as in characters) and typography (as in font and alignment).

class UIEventListener

This class declares a handler interface for user interface events.

· class Unicode

This class provides simple helper functions for working with 16-bit strings.

· class Vector

A very simple container class using pre-allocated memory.

class Vector4

This class represents a homogeneous 3D vector.

· class View

This is a generic touchgfx::Screen specialization for normal applications.

· class Widget

A Widget is a Drawable leaf (i.e. not a container).

class WildcardTextButtonStyle

A wildcard text button style.

class ZoomAnimationImage

Class for optimizing and wrapping move and zoom operations on ScalableImages.

Typedefs

typedef uint16 t Bitmapld

This type shall be used by the application to define unique IDs for all bitmaps in the system. The application shall define bitmap IDs in the range [0, number of bitmaps - 1].

typedef BoxWithBorderButtonStyle < ClickButtonTrigger > BoxClickButton

Defines an alias representing the box click button.

typedef BoxWithBorderButtonStyle < RepeatButtonTrigger > BoxRepeatButton

Defines an alias representing the box repeat button.

typedef BoxWithBorderButtonStyle < ToggleButtonTrigger > BoxToggleButton

Defines an alias representing the box toggle button.

• typedef BoxWithBorderButtonStyle < TouchButtonTrigger > BoxTouchButton

Defines an alias representing the box touch button.

typedef ImageButtonStyle < ClickButtonTrigger > ImageClickButton

Defines an alias representing the image click button.

• typedef ImageButtonStyle < RepeatButtonTrigger > ImageRepeatButton

Defines an alias representing the image repeat button.

typedef ImageButtonStyle < TouchButtonTrigger > ImageTouchButton

Defines an alias representing the image touch button.

typedef ImageButtonStyle < ToggleButtonTrigger > ImageToggleButton

Defines an alias representing the image toggle button.

• typedef IconButtonStyle < ClickButtonTrigger > IconClickButton

Defines an alias representing the icon click button.

typedef IconButtonStyle < RepeatButtonTrigger > IconRepeatButton

Defines an alias representing the icon repeat button.

typedef IconButtonStyle < TouchButtonTrigger > IconTouchButton

Defines an alias representing the icon touch button.

typedef lconButtonStyle < ToggleButtonTrigger > lconToggleButton

Defines an alias representing the icon toggle button.

typedef ImageButtonStyle < IconButtonStyle < ClickButtonTrigger > > IconImageClickButton

Defines an alias representing the icon image click button.

typedef ImageButtonStyle < IconButtonStyle < RepeatButtonTrigger > > IconImageRepeatButton

Defines an alias representing the icon image repeat button.

typedef ImageButtonStyle < IconButtonStyle < TouchButtonTrigger > > IconImageTouchButton

Defines an alias representing the icon image touch button.

• typedef ImageButtonStyle < IconButtonStyle < ToggleButtonTrigger >> IconImageToggleButton

Defines an alias representing the icon image toggle button.

 $\bullet \ \ type def \ TextButton Style < ClickButton Trigger > TextClickButton \\$

Defines an alias representing the text click button.

typedef TextButtonStyle < RepeatButtonTrigger > TextRepeatButton

Defines an alias representing the text repeat button.

typedef TextButtonStyle < TouchButtonTrigger > TextTouchButton

Defines an alias representing the text touch button.

typedef TextButtonStyle < ToggleButtonTrigger > TextToggleButton

Defines an alias representing the text toggle button.

 $\bullet \ \ type def \ TiledImage Button Style < Click Button Trigger > TiledImage Click Button \\$

Defines an alias representing the tiled image click button.

typedef TiledImageButtonStyle < RepeatButtonTrigger > TiledImageRepeatButton

Defines an alias representing the tiled image repeat button.

typedef TiledImageButtonStyle < TouchButtonTrigger > TiledImageTouchButton

Defines an alias representing the tiled image touch button.

- typedef TiledImageButtonStyle < ToggleButtonTrigger > TiledImageToggleButton
 Defines an alias representing the tiled image toggle button.
- typedef WildcardTextButtonStyle < ClickButtonTrigger > WildcardTextClickButton
 Defines an alias representing the wildcard text click button.
- typedef WildcardTextButtonStyle < RepeatButtonTrigger > WildcardTextRepeatButton
 Defines an alias representing the wildcard text repeat button.
- typedef WildcardTextButtonStyle < TouchButtonTrigger > WildcardTextTouchButton
 Defines an alias representing the wildcard text touch button.
- typedef WildcardTextButtonStyle < ToggleButtonTrigger > WildcardTextToggleButton
 Defines an alias representing the wildcard text toggle button.
- typedef TwoWildcardTextButtonStyle < ClickButtonTrigger > TwoWildcardTextClickButton
 Defines an alias representing the wildcard text click button.
- typedef TwoWildcardTextButtonStyle < RepeatButtonTrigger > TwoWildcardTextRepeatButton
 Defines an alias representing the wildcard text repeat button.
- typedef TwoWildcardTextButtonStyle < TouchButtonTrigger > TwoWildcardTextTouchButton
 Defines an alias representing the wildcard text touch button.
- typedef TwoWildcardTextButtonStyle < ToggleButtonTrigger > TwoWildcardTextToggleButton
 Defines an alias representing the wildcard text toggle button.
- typedef AnimatedImageButtonStyle < ClickButtonTrigger > AnimatedImageClickButton
 Defines an alias representing the animated image click button.
- typedef AnimatedImageButtonStyle < RepeatButtonTrigger > AnimatedImageRepeatButton
 Defines an alias representing the animated image repeat button.
- typedef AnimatedImageButtonStyle < TouchButtonTrigger > AnimatedImageTouchButton
 Defines an alias representing the animated image touch button.
- typedef AnimatedImageButtonStyle < ToggleButtonTrigger > AnimatedImageToggleButton
 Defines an alias representing the animated image toggle button.
- typedef int16_t(* EasingEquation) (uint16_t, int16_t, int16_t, uint16_t)
 This function pointer typedef matches the signature for all easing equations.
- typedef uint16 t FontId

Defines an alias representing identifier for the font.

• typedef uint8_t Alignment

Defines an alignment type.

• typedef uint8_t TextDirection

Defines a the direction to write text.

· typedef uint16 t RenderingVariant

Describes a combination of rendering algorithm, image format, and alpha information.

typedef int32_t fixed28_4

A fixed point value using 4 bits for the decimal part and 28 bits for the integral part.

typedef int32_t fixed16_16

A fixed point value using 16 bits for the decimal part and 16 bits for the integral part.

typedef uint16_t TypedTextId

Text IDs as generated by the text converter are simple uint16_t typedefs.

· typedef uint16 t Languageld

Language IDs generated by the text converter are uint16_t typedef'ed.

Enumerations

enum GlyphFlags { GLYPH_DATA_KERNINGTABLEPOS_BIT8_10 = 0x07, GLYPH_DATA_WIDTH_BIT8 = 0x08, GLYPH_DATA_HEIGHT_BIT8 = 0x10, GLYPH_DATA_TOP_BIT8 = 0x20, GLYPH_DATA_TOP_BIT9 = 0x40, GLYPH_DATA_ADVANCE_BIT8 = 0x80 }

Glyph flag definitions.

enum BlitOperations {

```
BLIT_OP_COPY = 1 << 0, BLIT_OP_FILL = 1 << 1, BLIT_OP_COPY_WITH_ALPHA = 1 << 2, BLIT_\hookleftarrow OP_FILL_WITH_ALPHA = 1 << 3, BLIT_OP_COPY_WITH_TRANSPARENT_PIXELS = 1 << 4, BLIT_\hookleftarrow OP_COPY_ARGB8888 = 1 << 5, BLIT_OP_COPY_ARGB8888_WITH_ALPHA = 1 << 6, BLIT_OP_COPY_A4 = 1 << 7, BLIT_OP_COPY_A8 = 1 << 8 }
```

The BlitOp operations.

• enum Direction { NORTH, SOUTH, EAST, WEST }

Defines a 2D direction type.

enum FrameBuffer { FB PRIMARY, FB SECONDARY, FB TERTIARY }

Defines a FrameBuffer type.

enum Gradient { GRADIENT HORIZONTAL, GRADIENT VERTICAL }

Defines a gradient type.

enum DisplayRotation { rotate0, rotate90 }

Defines a rotation of the display.

• enum DisplayOrientation { ORIENTATION LANDSCAPE, ORIENTATION PORTRAIT }

Defines the orientation of the display.

- enum TextRotation { TEXT_ROTATE_0, TEXT_ROTATE_90, TEXT_ROTATE_180, TEXT_ROTATE_270 } Defines a rotation of text.

Defines how long text lines should be dealt with if the width exceeds that of the TextArea.

• enum DMAType { DMA_TYPE_GENERIC, DMA_TYPE_CHROMART }

Defines a DMAType type.

Functions

• template < class HALType >

HAL & touchgfx_generic_init (DMA_Interface &dma, LCD &display, TouchController &tc, int16_t width, int16← _t height, uint16_t *bitmapCache, uint32_t bitmapCacheSize, uint32_t numberOfDynamicBitmaps=0)

TouchGFX generic initialize.

Prepare screen transition. Private helper function for makeTransition. Do not use.

• static void finalizeTransition (Screen *newScreen, Presenter *newPresenter, Transition *newTransition)

Finalize screen transition. Private helper function for make Transition. Do not use.

- template < class ScreenType , class PresenterType , class TransType , class ModelType >

PresenterType * makeTransition (Screen **currentScreen, Presenter **currentPresenter, MVPHeap &heap, Transition **currentTrans, ModelType *model)

Function for effectuating a screen transition (i.e. makes the requested new presenter/view pair active). Once this function has returned, the new screen has been transitioned to. Due to the memory allocation strategy of using the same memory area for all screens, the old view/presenter will no longer exist when this function returns.

• FORCE INLINE FUNCTION int LCD2shiftVal (int offset)

Shift value to get the right pixel in a byte.

• FORCE_INLINE_FUNCTION uint8_t LCD2getPixel (const uint8_t *addr, int offset)

Get pixel from buffer/image.

• FORCE_INLINE_FUNCTION uint8_t LCD2getPixel (const uint16_t *addr, int offset)

Get pixel from buffer/image.

```
6.1 touchgfx Namespace Reference

    FORCE_INLINE_FUNCTION void LCD2setPixel (uint8_t *addr, int offset, uint8_t value)

          Set pixel in buffer.
    • FORCE INLINE FUNCTION void LCD2setPixel (uint16 t *addr, int offset, uint8 t value)
          Set pixel in buffer.

    FORCE INLINE FUNCTION uint8 t LCD4getPixel (const uint8 t *addr, int offset)

          Get pixel from buffer/image.
    • FORCE INLINE FUNCTION uint8 t LCD4getPixel (const uint16 t *addr, int offset)
          Get pixel from buffer/image.
    • FORCE INLINE FUNCTION void LCD4setPixel (uint8 t *addr, int offset, uint8 t value)
          Set pixel in buffer.
    • FORCE_INLINE_FUNCTION void LCD4setPixel (uint16_t *addr, int offset, uint8_t value)
          Set pixel in buffer.
    void hw_init ()
          Function to perform generic hardware initialization of the board.
    void touchgfx_init ()
          Function to perform touchgfx initialization.

    void FrameBufferAllocatorWaitOnTransfer ()

          Called by FrameBufferAllocator to wait for a LCD Transfer.

    void FrameBufferAllocatorSignalBlockDrawn ()

          Called by FrameBufferAllocator when a block is drawn.

    Matrix4x4 operator* (const Matrix4x4 &multiplicand, const Matrix4x4 &multiplier)

          Multiplication operator.
    • Point4 operator* (const Matrix4x4 &multiplicand, const Point4 &multiplier)
          Multiplication operator.

    float fixed28 4ToFloat (fixed28 4 value)

          Fixed 28 4 to float.

    fixed28_4 floatToFixed28_4 (float value)

          Float to fixed 28 4.

    fixed16_16 floatToFixed16_16 (float value)

          Float to fixed 16.

    fixed28_4 fixed28_4Mul (fixed28_4 a, fixed28_4 b)

          Fixed 28 4 mul.

    int32 t ceil28 4 (fixed28 4 value)

          Ceiling 28 4.

    void floorDivMod (int32_t numerator, int32_t denominator, int32_t &floor, int32_t &mod)

          Floor div modifier.

    void memset (void *data, uint8 t c, uint32 t size)

          Simple implementation of the standard memset function.

    RenderingVariant lookupNearestNeighborRenderVariant (const Bitmap &bitmap)

          Returns the associated nearest neighbor render variant based on the bitmap format.
    • RenderingVariant lookupBilinearRenderVariant (const Bitmap &bitmap)
          Returns the associated bilinear render variant based on the bitmap format.

    template<typename T >

      Tabs (Td)
          Simple implementation of the standard abs function.

    template<typename T >

      T gcd (T a, T b)
          Find greatest common divisor.
```

• int32_t muldiv (int32_t factor1, int32_t factor2, int32_t divisor, int32_t &remainder)

 int32_t clz (int32_t x) Count leading zeros.

Multiply and divide.

Variables

const BitmapId BITMAP_ANIMATION_STORAGE = 0xFFFEU

A virtual id representing animation storage.

const BitmapId BITMAP_INVALID = 0xFFFFU

Define the bitmapld if an invalid bitmap.

• static const Alignment LEFT = 0

Text is left aligned.

static const Alignment CENTER = 1

Text is centered horizontally.

• static const Alignment RIGHT = 2

Text is right aligned.

static const TextDirection TEXT DIRECTION LTR = 0

Text is written Left-To-Right, e.g. English.

static const TextDirection TEXT_DIRECTION_RTL = 1

Text is written Right-To-Left, e.g. Hebrew.

• static const uint16_t RenderingVariant_NearestNeighbor = 0

The rendering variant nearest neighbor bit value.

static const uint16_t RenderingVariant_Bilinear = 1

The rendering variant bilinear bit value.

• static const uint16_t RenderingVariant_NoAlpha = 0

The rendering variant no alpha bit value.

static const uint16_t RenderingVariant_Alpha = 2

The rendering variant alpha bit value.

static const uint16_t RenderingVariant_FormatShift = 2

The rendering variant format shift.

• static const float PI = 3.14159265358979323846f

PI.

const TypedTextId TYPED_TEXT_INVALID = 0xFFFFU

The ID of an invalid text.

6.1.1 Typedef Documentation

6.1.1.1 Alignment

uint8_t Alignment

Defines an alignment type.

6.1.1.2 EasingEquation

```
int16_t(* EasingEquation)(uint16_t, int16_t, int16_t, uint16_t)
```

This function pointer typedef matches the signature for all easing equations. Thereby EasingEquation is a convenient shorthand for a pointer to any easing equation.

6.1.1.3 Rendering Variant

uint16_t RenderingVariant

Describes a combination of rendering algorithm, image format, and alpha information. The lowest bit is 0 for " \leftarrow Nearest neighbor", 1 for "Bilinear". The next bit is "0" for "no alpha", "2" for "alpha". The rest is the Bitmap::Format shifted up by 2.

6.1.1.4 TextDirection

uint8_t TextDirection

Defines a the direction to write text.

6.1.2 Enumeration Type Documentation

6.1.2.1 BlitOperations

enum BlitOperations

The BlitOp operations.

Enumerator

BLIT_OP_COPY	Copy the source to the destination.
BLIT_OP_FILL	Fill the destination with color.
BLIT_OP_COPY_WITH_ALPHA	Copy the source to the destination using the given alpha.
BLIT_OP_FILL_WITH_ALPHA	Fill the destination with color using the given alpha.
BLIT_OP_COPY_WITH_TRANSPARENT_PIXELS	Deprecated, ignored. (Copy the source to the destination, but not the transparent pixels)
BLIT_OP_COPY_ARGB8888	Copy the source to the destination, performing per-pixel alpha blending.
BLIT_OP_COPY_ARGB8888_WITH_ALPHA	Copy the source to the destination, performing per-pixel alpha blending and blending the result with an image-wide alpha.
BLIT_OP_COPY_A4	Copy 4-bit source text to destination, performing per-pixel alpha blending.
BLIT_OP_COPY_A8	Copy 8-bit source text to destination, performing per-pixel alpha blending.

6.1.2.2 Direction

enum enum Direction

Defines a 2D direction type.

Enumerator

NO	ORTH	An enum constant representing the north option.
SC	HTUC	An enum constant representing the south option.

Enumerator

EAST	An enum constant representing the east option.
WEST	An enum constant representing the west option.

6.1.2.3 DisplayOrientation

enum enum DisplayOrientation

Defines the orientation of the display.

Enumerator

ORIENTATION_LANDSCAPE	The display has more pixels from left to right than from top to bottom.
ORIENTATION_PORTRAIT	The display has more pixels from top to bottom than from right to left.

6.1.2.4 DisplayRotation

enum enum DisplayRotation

Defines a rotation of the display.

See also

DisplayOrientation

Enumerator

rotate0	The display is oriented like the frame buffer.]
rotate90	The display is rotated 90 degrees compared to the frame buffer layout.]

6.1.2.5 DMAType

enum enum DMAType

Defines a DMAType type.

Enumerator

DMA_TYPE_GENERIC	Generic DMA Implementation.
DMA_TYPE_CHROMART	ChromART hardware DMA Implementation.

6.1.2.6 FrameBuffer

 $\verb"enum" enum FrameBuffer"$

Defines a FrameBuffer type.

Enumerator

FB_PRIMARY	First frame buffer.
FB_SECONDARY	Second frame buffer.
FB_TERTIARY	Third frame buffer.

6.1.2.7 GlyphFlags

enum GlyphFlags

Glyph flag definitions.

Enumerator

GLYPH_DATA_KERNINGTABLEPOS_BIT8_10	The 8th, 9th and 10th bit of the kerningTablePos.
GLYPH_DATA_WIDTH_BIT8	The 9th bit of "width".
GLYPH_DATA_HEIGHT_BIT8	The 9th bit of "height".
GLYPH_DATA_TOP_BIT8	The 9th bit of "top".
GLYPH_DATA_TOP_BIT9	The sign bit of "top".
GLYPH_DATA_ADVANCE_BIT8	The 9th bit of "advance".

6.1.2.8 Gradient

enum enum Gradient

Defines a gradient type.

Enumerator

GRADIENT_HORIZONTAL	Horizontal gradient.
GRADIENT_VERTICAL	Vertical gradient.

6.1.2.9 TextRotation

enum enum TextRotation

Defines a rotation of text. Each enumeration option specifies the number of degrees the text is turned clockwise.

Enumerator

TEXT_ROTATE_0	Text is written from left to right.
TEXT_ROTATE_90	Text is written from top to bottom.
TEXT_ROTATE_180	Text is written from right to left (upside down)
TEXT ROTATE 270	Text is written bottom to top.

6.1.2.10 WideTextAction

enum enum WideTextAction

See also

TextArea::setWideTextAction

Enumerator

WIDE_TEXT_NONE	Do nothing, simply cut the text in the middle of any character that extends beyond the width of the TextArea.
WIDE_TEXT_WORDWRAP	Wrap between words, ellipsis anywhere "Very long t".
WIDE_TEXT_WORDWRAP_ELLIPSIS_AFTER_S↔ PACE	Wrap between words, ellipsis anywhere only after space "Very long".
WIDE_TEXT_CHARWRAP	Wrap between any two characters, ellipsis anywhere, as used in Chinese.
WIDE_TEXT_CHARWRAP_DOUBLE_ELLIPSIS	Wrap between any two characters, double ellipsis anywhere, as used in Chinese.

6.1.3 Function Documentation

6.1.3.1 abs()

```
template< class T > T abs (
```

Simple implementation of the standard abs function.

Template Parameters

Т	The type on which to perform the abs.
	, ,,

Parameters

d The entity on which to perform the abs.

Returns

The absolute (non-negative) value of d.

6.1.3.2 ceil28_4()

Parameters

value	The value.
-------	------------

Returns

The ceil result.

6.1.3.3 clz()

```
static int32_t clz ( int32\_t \ x \ )
```

Count leading zeros in the binary representation of the given value.

Parameters

```
x The value to count the number of leading zeros in.
```

Returns

An int32_t.

6.1.3.4 finalizeTransition()

Parameters

in	newScreen	If non-null, the new screen.
in	newPresenter	If non-null, the new presenter.
in	newTransition	If non-null, the new transition.

6.1.3.5 fixed28_4Mul()

Parameters

а	The fixed28_4 to process.
b	The fixed28_4 to process.

Returns

the result.

6.1.3.6 fixed28_4ToFloat()

Parameters

Returns

The value as float.

6.1.3.7 floatToFixed16_16()

Parameters

value .	The value.
---------	------------

Returns

The value as fixed16_16.

6.1.3.8 floatToFixed28_4()

Parameters

```
value The value.
```

Returns

The value as fixed28_4.

6.1.3.9 floorDivMod()

```
int32_t denominator,
int32_t & floor,
int32_t & mod ) [inline]
```

Parameters

	numerator	The numerator.
	denominator	The denominator.
in,out	floor	The floor.
in,out	mod	The modifier.

6.1.3.10 FrameBufferAllocatorSignalBlockDrawn()

```
void FrameBufferAllocatorSignalBlockDrawn ( )
```

Called by FrameBufferAllocator when a block is drawn and therefore ready for transfer. The LCD driver should use this method to start a transfer.

6.1.3.11 FrameBufferAllocatorWaitOnTransfer()

```
void FrameBufferAllocatorWaitOnTransfer ( )
```

Called by FrameBufferAllocator to wait for a LCD Transfer, when the allocator has no free blocks. The LCD driver can use this function to synchronize the UI thread with the transfer logic.

6.1.3.12 gcd()

```
template< typename T > T gcd ( T a, T b )
```

Find greatest common divisor of two given numbers.

Template Parameters

T Generic type parameter.

Parameters

а	The first number.
b	The second number.

Returns

AT.

6.1.3.13 hw_init()

```
void hw_init ( )
```

Function to perform generic hardware initialization of the board. This function prototype is only provided as a convention.

6.1.3.14 LCD2getPixel() [1/2]

Parameters

addr	The address.
offset	The offset.

Returns

The pixel value.

6.1.3.15 LCD2getPixel() [2/2]

Parameters

addr	The address.
offset	The offset.

Returns

The pixel value.

6.1.3.16 LCD2setPixel() [1/2]

Parameters

in	addr	The address.
	offset	The offset.
	value	The value.

6.1.3.17 LCD2setPixel() [2/2]

uint8_t value)

Parameters

in	addr	The address.
	offset	The offset.
	value	The value.

6.1.3.18 LCD2shiftVal()

```
\begin{tabular}{ll} FORCE\_INLINE\_FUNCTION int LCD2shiftVal ( \\ & int offset ) \end{tabular}
```

Parameters

Returns

The shift value.

6.1.3.19 LCD4getPixel() [1/2]

Parameters

addr	The address.
offset	The offset.

Returns

The pixel value.

6.1.3.20 LCD4getPixel() [2/2]

Parameters

addr	The address.
offset	The offset.

Returns

The pixel value.

6.1.3.21 LCD4setPixel() [1/2]

Parameters

in	addr	The address.
	offset	The offset.
	value	The value.

6.1.3.22 LCD4setPixel() [2/2]

Parameters

in	addr	The address.
	offset	The offset.
	value	The value.

6.1.3.23 lookupBilinearRenderVariant()

Returns the associated bilinear render variant based on the bitmap format.

Parameters

bitmap	The bitmap.

Returns

A Rendering Variant based on the bitmap format.

6.1.3.24 lookupNearestNeighborRenderVariant()

RenderingVariant lookupNearestNeighborRenderVariant (

```
const Bitmap & bitmap )
```

Returns the associated nearest neighbor render variant based on the bitmap format.

Parameters

bitmap The bitmap.

Returns

A Rendering Variant based on the bitmap format.

6.1.3.25 makeTransition()

Will properly clean up old screen (tearDownScreen, Presenter::deactivate) and call setupScreen/activate on new view/presenter pair. Will also make sure the view, presenter and model are correctly bound to each other.

Template Parameters

ScreenType	Class type for the View.
	1 1
PresenterType	Class type for the Presenter.
TransType	Class type for the Transition.
ModelType	Class type for the Model.

Parameters

in	currentScreen	Pointer to pointer to the current view.
in	in currentPresenter Pointer to pointer to the current presenter.	
in	heap	Reference to the heap containing the memory storage in which to allocate.
in	currentTrans	Pointer to pointer to the current transition.
in	model	Pointer to model.

Returns

Pointer to the new Presenter of the requested type. Incidentally it will be the same value as the old presenter due to memory reuse.

6.1.3.26 memset()

```
void memset (
    void * data,
    uint8_t c,
    uint32_t size )
```

Provides utility functions.

Simple implementation of the standard memset function.

Parameters

out	data	lata Address of data to set.	
	С	Value to set.	
	size	Number of bytes to set.	

6.1.3.27 muldiv()

Multiply and divide without causing overflow. Multiplying two large values and subsequently dividing the result with another large value might cause an overflow in the intermediate result. The function muldiv() will multiply the two first values and divide the result by the third value without causing overflow (unless the final result would overflow). The remainder from the calculation is also returned.

Parameters

	factor1	The first factor.
factor2		The second factor.
	divisor	The divisor.
out	remainder	The remainder.

Returns

An int32_t.

Multiplication operator.

Parameters

multiplicand	The first value to multiply.
multiplier	The second value to multiply.

Returns

The result of the operation.

Multiplication operator.

Parameters

multiplicand	The first value to multiply.
multiplier	The second value to multiply.

Returns

The result of the operation.

6.1.3.30 prepareTransition()

Parameters

in	currentScreen	If non-null, the current screen.
in	currentPresenter	If non-null, the current presenter.
in	currentTrans	If non-null, the current transaction.

6.1.3.31 touchgfx_generic_init()

```
template< class HALType > HAL & touchgfx_generic_init (
    DMA_Interface & dma,
    LCD & display,
    TouchController & tc,
    int16_t width,
    int16_t height,
    uint16_t * bitmapCache,
    uint32_t bitmapCacheSize,
    uint32_t numberOfDynamicBitmaps = 0 )
```

Functions

TouchGFX generic initialize.

Template Parameters

HALType The class type of the HAL subclass used for this port.

Parameters

in	dma	Reference to the DMA implementation object to use. Can be of type NoDMA to disable the use of DMA for rendering.
in	display	Reference to the LCD renderer implementation (subclass of LCD). Could be either LCD16bpp for RGB565 Uls, or LCD1bpp for monochrome Uls or LCD24bpp for 24bit displays using RGB888 Uls.
in	tc	Reference to the touch controller driver (or NoTouchController to disable touch input).
	width	The <i>native</i> display width of the actual display, in pixels. This value is irrespective of whether the concrete UI should be portrait or landscape mode. It must match what the display itself is configured as.
	height	The <i>native</i> display height of the actual display, in pixels. This value is irrespective of whether the concrete UI should be portrait or landscape mode. It must match what the display itself is configured as.
in	bitmapCache	Optional pointer to starting address of a memory region in which to place the bitmap cache. Usually in external RAM. Pass 0 if bitmap caching is not used.
	bitmapCacheSize	Size of bitmap cache in bytes. Pass 0 if bitmap cache is not used.
	numberOfDynamicBitmaps	Number of dynamic bitmaps.

Returns

A reference to the allocated (and initialized) HAL object.

6.1.3.32 touchgfx_init()

```
void touchgfx_init ( )
```

Function to perform touchgfx initialization. This function prototype is only provided as a convention.

6.1.4 Variable Documentation

6.1.4.1 TYPED_TEXT_INVALID

```
const TypedTextId TYPED_TEXT_INVALID = 0xFFFFU
```

This type shall be used by the application to define unique IDs for all typed texts in the system. The application shall define typed text IDs in the range [0,number of typed texts - 1].

Chapter 7

Class Documentation

7.1 AbstractButton Class Reference

This class defines an abstract interface for button-like elements.

#include <touchgfx/widgets/AbstractButton.hpp>

Public Member Functions

AbstractButton ()

Constructor.

· virtual void handleClickEvent (const ClickEvent &event)

Updates the current state of the button.

void setAction (GenericCallback< const AbstractButton & > &callback)

Associates an action to be performed when the AbstractButton is clicked.

• virtual bool getPressedState () const

Function to determine whether this AbstractButton is currently pressed.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

GenericCallback< const AbstractButton &> * action

The callback to be executed when this AbstractButton is clicked.

· bool pressed

Is the button pressed or released? True if pressed.

Additional Inherited Members

7.1.1 Detailed Description

This class defines an abstract interface for button-like elements. A button is a clickable element that has two states - pressed or released - and executes an action when the pressed->released transition is made.

See also

Widget

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AbstractButton()

```
AbstractButton ( ) [inline]
```

Constructs an AbstractButton instance in released state without an associated action.

7.1.3 Member Function Documentation

7.1.3.1 getPressedState()

```
bool getPressedState ( ) const [inline], [virtual]
```

Function to determine whether this AbstractButton is currently pressed.

Returns

true if button is pressed, false otherwise.

7.1.3.2 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_ABSTRACTBUTTON.

Reimplemented from Widget.

Reimplemented in RadioButton, ButtonWithLabel, ButtonWithlcon, Button, TouchArea, and ToggleButton.

7.1.3.3 handleClickEvent()

Updates the current state of the button - pressed or released - and invalidates it.

If a transition from the pressed to the released state was made, the associated action is executed and then the Widget is invalidated.

Parameters

event	Information about the click.
-------	------------------------------

See also

Drawable::handleClickEvent()

Reimplemented from Drawable.

Reimplemented in RepeatButton, ToggleButton, RadioButton, and TouchArea.

7.1.3.4 setAction()

Associates an action to be performed when the AbstractButton is clicked.

Parameters

callback The callback to be executed. The callback will be given a reference to the AbstractButton.

See also

GenericCallback

7.2 AbstractButtonContainer Class Reference

An abstract button container.

#include <touchgfx/containers/buttons/AbstractButtonContainer.hpp>

Public Member Functions

• AbstractButtonContainer ()

Default constructor.

virtual ~AbstractButtonContainer ()

Destructor.

• void setPressed (bool isPressed)

Sets the pressed state.

• bool getPressed ()

Gets the pressed state.

void setAlpha (uint8_t newAlpha)

Sets an alpha value.

• uint8_t getAlpha () const

Gets the alpha.

void setAction (GenericCallback< const AbstractButtonContainer & > &callback)

Sets an action callback.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

· virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· bool pressed

True if pressed.

• uint8_t alpha

The current alpha value. 255 denotes solid, 0 denotes completely transparent.

GenericCallback< const AbstractButtonContainer &> * action

The action.

Additional Inherited Members

7.2.1 Detailed Description

An abstract button container. It defines pressed/not pressed state, the alpha value, and the action Callback of a button. AbstractButtonContainer is used as superclass for classes defining a specific button behaviour.

See also

ClickButtonTrigger, RepeatButtonTrigger, ToggleButtonTrigger, TouchButtonTrigger

7.2.2 Member Function Documentation

```
7.2.2.1 getAlpha()
```

```
uint8_t getAlpha ( ) const [inline]
```

Returns

The alpha value.

7.2.2.2 getPressed()

```
bool getPressed ( ) [inline]
```

Returns

True if it succeeds, false if it fails.

7.2.2.3 setAction()

Parameters

callback	The callback.
calibach	THE Callback.

7.2.2.4 setAlpha()

Parameters

7.2.2.5 setPressed()

Parameters

7.3 AbstractClock Class Reference

Superclass of clock widgets.

```
#include <touchgfx/containers/clock/AbstractClock.hpp>
```

Public Member Functions

• AbstractClock ()

Default constructor.

virtual ∼AbstractClock ()

Destructor.

• virtual void setTime24Hour (uint8_t hour, uint8_t minute, uint8_t second)

Sets the time with input format as 24H.

• virtual void setTime12Hour (uint8_t hour, uint8_t minute, uint8_t second, bool am)

Sets the time with input format as 12H.

• uint8_t getCurrentHour () const

Gets the current hour.

• uint8_t getCurrentMinute () const

Gets the current minute.

• uint8_t getCurrentSecond () const

Gets the current second.

Protected Member Functions

• virtual void updateClock ()=0

Updates the visual representation of the clock.

Protected Attributes

• uint8_t currentHour

Local copy of the current hour.

• uint8_t currentMinute

Local copy of the current minute.

• uint8_t currentSecond

Local copy of the current second.

Additional Inherited Members

7.3.1 Constructor & Destructor Documentation

```
7.3.1.1 AbstractClock()
```

```
AbstractClock ( )
```

Default constructor.

```
7.3.1.2 ~AbstractClock()
```

```
~AbstractClock ( ) [inline], [virtual]
```

Destructor.

7.3.2 Member Function Documentation

```
7.3.2.1 getCurrentHour()
```

```
uint8_t getCurrentHour ( ) const
```

Gets the current hour.

Returns

The current hour.

7.3.2.2 getCurrentMinute()

```
uint8_t getCurrentMinute ( ) const
```

Gets the current minute.

Returns

The current minute.

7.3.2.3 getCurrentSecond()

```
uint8_t getCurrentSecond ( ) const
```

Gets the current second.

Returns

The current second.

7.3.2.4 setTime12Hour()

Sets the time with input format as 12H. Note that this does not affect any selected presentation formats.

Parameters

hour	The hours (in 12H format).
minute	The minutes (in 12H format).
second	The seconds (in 12H format).
am	AM/PM setting. True = AM, false = PM.

7.3.2.5 setTime24Hour()

Sets the time with input format as 24H. Note that this does not affect any selected presentation formats.

Parameters

hour	The hours (in 24H format).
minute	The minutes (in 24H format).
second	The seconds (in 24H format).

7.3.2.6 updateClock()

```
void updateClock ( ) [protected], [pure virtual]
```

Updates the visual representation of the clock.

Implemented in AnalogClock, and DigitalClock.

7.4 AbstractDirectionProgress Class Reference

An abstract direction progress.

 $\verb|#include| < touchgfx/containers/progress_indicators/AbstractDirectionProgress. \leftarrow | hpp >$

Public Types

enum DirectionType { RIGHT, LEFT, DOWN, UP }

Values that represent directions.

Public Member Functions

AbstractDirectionProgress ()

Default constructor.

virtual ~AbstractDirectionProgress ()

Destructor.

virtual void setDirection (DirectionType direction)

Sets a direction.

virtual DirectionType getDirection () const

Gets the direction.

Protected Attributes

• DirectionType progressDirection

The progress direction.

Additional Inherited Members

7.4.1 Detailed Description

An abstract direction progress for progress indicators that need a direction to be specified.

7.4.2 Member Enumeration Documentation

7.4.2.1 DirectionType

enum enum DirectionType

Values that represent directions.

7.4.3 Constructor & Destructor Documentation

7.4.3.1 AbstractDirectionProgress()

```
AbstractDirectionProgress ( )
```

Default constructor.

7.4.3.2 ~AbstractDirectionProgress()

```
\simAbstractDirectionProgress ( ) [virtual]
```

Destructor.

7.4.4 Member Function Documentation

7.4.4.1 getDirection()

```
DirectionType getDirection ( ) const [virtual]
```

Gets the direction.

Returns

The direction.

7.4.4.2 setDirection()

Sets a direction.

Parameters

direction The direction.

7.5 AbstractPainter Class Reference

An abstract class for creating painter classes for drawing canvas widgets.

```
#include <touchgfx/widgets/canvas/AbstractPainter.hpp>
```

Public Member Functions

AbstractPainter ()

Default constructor.

virtual ∼AbstractPainter ()

Destructor.

void setOffset (uint16 t offsetX, uint16 t offsetY)

Sets the offset of the area being drawn.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)=0

Paint a designated part of the RenderingBuffer.

Protected Member Functions

void setWidgetAlpha (uint8_t alpha)
 Sets widget alpha.

Static Protected Member Functions

• static bool compatibleFramebuffer (Bitmap::BitmapFormat format)

Check if the provided bitmap format matches the current framebuffer format.

Protected Attributes

int16_t areaOffsetX

The offset x coordinate of the area being drawn.

· int16 tareaOffsetY

The offset y coordinate of the area being drawn.

uint8_t widgetAlpha

The alpha of the widget using the painter.

7.5.1 Detailed Description

An abstract class for creating painter classes for drawing canvas widgets.

7.5.2 Constructor & Destructor Documentation

```
7.5.2.1 AbstractPainter()
```

```
AbstractPainter ( )
```

Default constructor.

7.5.2.2 ∼AbstractPainter()

```
\simAbstractPainter ( ) [virtual]
```

Destructor.

7.5.3 Member Function Documentation

7.5.3.1 compatibleFramebuffer()

Helper function to check if the provided bitmap format matches the current framebuffer format.

Parameters

format	A bitmap format.
--------	------------------

Returns

True if the format matches the framebuffer format, false otherwise.

7.5.3.2 render()

```
void render (
     uint8_t * ptr,
     int x,
     int xAdjust,
     int y,
     unsigned count,
     const uint8_t * covers ) [pure virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Χ	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implemented in PainterARGB8888, PainterGRAY2, PainterGRAY4, PainterRGB565, PainterRGB888, Painter← ABGR2222, PainterARGB2222, PainterBGRA2222, PainterARGB8888Bitmap, PainterARG← B8888L8Bitmap, PainterGRAY2Bitmap, PainterGRAY4Bitmap, PainterRGB565Bitmap, PainterRGB565L8Bitmap, PainterRGB888Bitmap, PainterABGR2222Bitmap, PainterARGB2222Bitmap, PainterABGR2222Bitmap, PainterBGRA222Bitmap, PainterABGR222Bitmap, PainterABGR22Bitmap, Pa

7.5.3.3 setOffset()

Sets the offset of the area being drawn. This allows render() to calculate the x, y relative to the widget, and not just relative to the invalidated area.

Parameters

offsetX	The offset x coordinate of the invalidated area relative to the widget.
offsetY	The offset y coordinate of the invalidated area relative to the widget.

7.5.3.4 setWidgetAlpha()

Sets the widget alpha to allow an entire canvas widget to easily be faded without changing the painter of the widget.

Parameters

alpha	The alpha.
-------	------------

Note

Used internally by Canvas.

7.6 AbstractPainterABGR2222 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/AbstractPainterABGR2222.hpp>
```

Public Member Functions

- virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

 Paint a designated part of the RenderingBuffer.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t newpix, uint8_t bufpix, uint8_t alpha)
 Mix colors.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t R, uint8_t G, uint8_t B, uint8_t bufpix, uint8_t alpha)
 Mix colors.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0
 - Get the color of the next pixel in the scan line.
- virtual void renderPixel (uint8_t *p, uint8_t red, uint8_t green, uint8_t blue)
 Renders the pixel.

Protected Attributes

int currentX

Current x coordinate relative to the widget.

· int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.6.1 Detailed Description

The AbstractPainterABGR2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.6.2 Member Function Documentation

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

newpix	The newpix value.
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.6.2.2 mixColors() [2/2]

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

R	The red color (0-255).
G	The green color (0-255).
В	The blue color (0-255).
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.6.2.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Χ	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterABGR2222, and PainterABGR2222Bitmap.

7.6.2.4 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterABGR2222Bitmap.

7.6.2.5 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterABGR2222, and PainterABGR2222Bitmap.

7.6.2.6 renderPixel()

```
void renderPixel (
          uint8_t * p,
          uint8_t red,
          uint8_t green,
          uint8_t blue ) [protected], [virtual]
```

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 2,2,2.

Parameters

in	р	pointer into the frame buffer where the given color should be written.	
	red	The red color.	
	green	The green color.	
	blue	The blue color.	

7.7 AbstractPainterARGB2222 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/AbstractPainterARGB2222.hpp>
```

Public Member Functions

- virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

 Paint a designated part of the RenderingBuffer.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t newpix, uint8_t bufpix, uint8_t alpha)
 Mix colors.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t R, uint8_t G, uint8_t B, uint8_t bufpix, uint8_t alpha) Mix colors.

Protected Member Functions

- virtual bool renderInit ()
 - Initialize rendering of a single scan line of pixels for the render.
- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0

Get the color of the next pixel in the scan line.

• virtual void renderPixel (uint8_t *p, uint8_t red, uint8_t green, uint8_t blue)

Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

· int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.7.1 Detailed Description

The AbstractPainterARGB2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.7.2 Member Function Documentation

```
7.7.2.1 mixColors() [1/2]
```

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

newpix The newpix value.	
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.7.2.2 mixColors() [2/2]

```
uint8_t bufpix,
uint8_t alpha ) [inline]
```

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

R	The red color (0-255).		
G	The green color (0-255).		
В	The blue color (0-255).		
bufpix	The bufpix value.		
alpha	The alpha of the newpix.		

Returns

The new color to write to the frame buffer.

7.7.2.3 render()

```
virtual void render (
            uint8_t * ptr,
            int x,
            int xAdjust,
            int y,
            unsigned count,
            const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterARGB2222, and PainterARGB2222Bitmap.

7.7.2.4 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterARGB2222Bitmap.

7.7.2.5 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

 $Implemented\ in\ Painter ARGB2222,\ and\ Painter ARGB2222 Bitmap.$

7.7.2.6 renderPixel()

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 2,2,2.

Parameters

in	р	pointer into the frame buffer where the given color should be written.
	red	The red color.
	green	The green color.
	blue	The blue color.

7.8 AbstractPainterARGB8888 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/AbstractPainterARGB8888.hpp>

Public Member Functions

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0

 Get the color of the next pixel in the scan line.
- virtual void renderPixel (uint16_t *p, uint8_t red, uint8_t green, uint8_t blue)
 Renders the pixel.
- virtual void renderPixel (uint16_t *p, uint8_t red, uint8_t green, uint8_t blue, uint8_t alpha)
 Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.8.1 Detailed Description

The AbstractPainterARGB8888 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.8.2 Member Function Documentation

7.8.2.1 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
----	-----	--

Parameters

	X	The x coordinate.	
xAdjust The minor adjustment of x (used when a pixel is smaller than a byte to specify that the should have been advanced "xAdjust" pixels futher).		The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	y The y coordinate.		
count Number of pixels to fill.		Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterARGB8888, PainterARGB8888Bitmap, and PainterARGB8888L8Bitmap.

7.8.2.2 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterARGB8888Bitmap, and PainterARGB8888L8Bitmap.

7.8.2.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterARGB8888, PainterARGB8888Bitmap, and PainterARGB8888L8Bitmap.

```
7.8.2.4 renderPixel() [1/2]
```

```
uint8_t red,
uint8_t green,
uint8_t blue ) [protected], [virtual]
```

Renders the pixel into the frame buffer.

Parameters

in	р	pointer into the frame buffer where the given color should be written.	
	red	The red color.	
	green	The green color.	
	blue	The blue color.	

Renders the pixel into the frame buffer.

Parameters

in	р	pointer into the frame buffer where the given color should be written.		
	red	The red color.		
	green	The green color.		
	blue	The blue color.		
	alpha	The alpha.		

7.9 AbstractPainterBGRA2222 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/AbstractPainterBGRA2222.hpp>
```

Public Member Functions

- virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

 Paint a designated part of the RenderingBuffer.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t newpix, uint8_t bufpix, uint8_t alpha)
 Mix colors.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t R, uint8_t G, uint8_t B, uint8_t bufpix, uint8_t alpha)

 Mix colors.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0

Get the color of the next pixel in the scan line.

• virtual void renderPixel (uint8_t *p, uint8_t red, uint8_t green, uint8_t blue)

Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

· int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.9.1 Detailed Description

The AbstractPainterBGRA2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.9.2 Member Function Documentation

```
7.9.2.1 mixColors() [1/2]
```

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

newpix	The newpix value.
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.9.2.2 mixColors() [2/2]

```
uint8_t G,
uint8_t B,
uint8_t bufpix,
uint8_t alpha ) [inline]
```

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

R	The red color (0-255).	
G	The green color (0-255).	
В	The blue color (0-255).	
bufpix	The bufpix value.	
alpha	The alpha of the newpix.	

Returns

The new color to write to the frame buffer.

7.9.2.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	Idjust The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointe should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterBGRA2222, and PainterBGRA2222Bitmap.

7.9.2.4 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterBGRA2222Bitmap.

7.9.2.5 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterBGRA2222, and PainterBGRA2222Bitmap.

7.9.2.6 renderPixel()

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 2,2,2.

Parameters

in	р	pointer into the frame buffer where the given color should be written.	
	red	The red color.	
	green	The green color.	
	blue	The blue color.	

7.10 AbstractPainterBW Class Reference

A Painter that will paint using a color on a LCD1bpp display.

#include <touchgfx/widgets/canvas/AbstractPainterBW.hpp>

Public Member Functions

 virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers) Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &color)=0

Get the color of the next pixel in the scan line.

Protected Attributes

uint16_t currentX

Current x coordinate relative to the widget.

uint16_t currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.10.1 Detailed Description

AbstractPainterBW is used for drawing one 1bpp displays. The color is either on or off No transparency is supported.

See also

AbstractPainter

7.10.2 Member Function Documentation

7.10.2.1 render()

```
virtual void render (
             uint8_t * ptr,
             int x_{\bullet}
             int xAdjust,
             int y,
             unsigned count,
             const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
TouchCi	covers	The coverage in of each pixel.	

uchGFX v4.12.3 (C)opyright STMicroelectronics 2014-2019

Implements AbstractPainter.

Reimplemented in PainterBW, and PainterBWBitmap.

7.10.2.2 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterBWBitmap.

7.10.2.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

C	out	color	Color of the pixel, 0 or 1.
---	-----	-------	-----------------------------

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterBW, and PainterBWBitmap.

7.11 AbstractPainterGRAY2 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/AbstractPainterGRAY2.hpp>
```

Public Member Functions

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &gray, uint8_t &alpha)=0

Get the color of the next pixel in the scan line.

virtual void renderPixel (uint8_t *p, uint16_t offset, uint8_t gray)

Renders the pixel.

Protected Attributes

int currentX

Current x coordinate relative to the widget.

int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.11.1 Detailed Description

The AbstractPainterGRAY2 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.11.2 Member Function Documentation

7.11.2.1 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterGRAY2, and PainterGRAY2Bitmap.

7.11.2.2 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterGRAY2Bitmap.

7.11.2.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	gray	The gray (0-15).
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterGRAY2, and PainterGRAY2Bitmap.

7.11.2.4 renderPixel()

Renders the pixel into the frame buffer.

Parameters

in	р	pointer into the frame buffer line where the given pixel should be written.	
	offset	The offset to the pixel from the given pointer.	
	gray	The green color.	

param blue The blue color.

7.12 AbstractPainterGRAY4 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/AbstractPainterGRAY4.hpp>

Public Member Functions

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &gray, uint8_t &alpha)=0

Get the color of the next pixel in the scan line.

 $\bullet \ \ virtual \ void \ \underline{renderPixel} \ (uint8_t \ *p, \ uint16_t \ offset, \ uint8_t \ gray)\\$

Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.12.1 Detailed Description

The AbstractPainterGRAY4 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.12.2 Member Function Documentation

7.12.2.1 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	X	The x coordinate.	

Parameters

xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
У	The y coordinate.	
count	Number of pixels to fill.	
covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterGRAY4, and PainterGRAY4Bitmap.

7.12.2.2 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterGRAY4Bitmap.

7.12.2.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	gray	The gray (0-15).
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterGRAY4, and PainterGRAY4Bitmap.

7.12.2.4 renderPixel()

Renders the pixel into the frame buffer.

Parameters

in	р	pointer into the frame buffer line where the given pixel should be written.	
	offset	The offset to the pixel from the given pointer.	
	gray	The green color.	

param blue The blue color.

7.13 AbstractPainterRGB565 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/AbstractPainterRGB565.hpp>

Public Member Functions

- virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

 Paint a designated part of the RenderingBuffer.
- FORCE_INLINE_FUNCTION uint16_t mixColors (uint16_t newpix, uint16_t bufpix, uint8_t alpha)
 Mix colors.
- FORCE_INLINE_FUNCTION uint16_t mixColors (uint16_t R, uint16_t G, uint16_t B, uint16_t bufpix, uint8_t alpha)

Mix colors.

Static Public Attributes

static const uint16_t RMASK = 0xF800

Mask for red (111110000000000)

• static const uint16_t GMASK = 0x07E0

Mask for green (00000111111100000)

static const uint16_t BMASK = 0x001F

Mask for blue (000000000011111)

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0
 - Get the color of the next pixel in the scan line.
- virtual void renderPixel (uint16_t *p, uint8_t red, uint8_t green, uint8_t blue)

Renders the pixel.

Protected Attributes

int currentX

Current x coordinate relative to the widget.

int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.13.1 Detailed Description

The AbstractPainterRGB565 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.13.2 Member Function Documentation

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

newpix	The newpix value.	
bufpix	The bufpix value.	
alpha	The alpha of the newpix.	

Returns

The new color to write to the frame buffer.

7.13.2.2 mixColors() [2/2]

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

R	The red color (placed in 0xF800).
G	The green color (placed in 0x03E0).
В	The blue color (placed in 0x001F).
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.13.2.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	x The x coordinate.	
	xAdjust The minor adjustment of x (used when a pixel is smaller than a byte to specify that the point should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Implements AbstractPainter.

Reimplemented in PainterRGB565, PainterRGB565Bitmap, and PainterRGB565L8Bitmap.

7.13.2.4 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterRGB565Bitmap, and PainterRGB565L8Bitmap.

7.13.2.5 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterRGB565, PainterRGB565Bitmap, and PainterRGB565L8Bitmap.

7.13.2.6 renderPixel()

```
void renderPixel (
          uint16_t * p,
          uint8_t red,
          uint8_t green,
          uint8_t blue ) [protected], [virtual]
```

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 5,6, 5.

Parameters

ĺ	in	р	pointer into the frame buffer where the given color should be written.	
ſ		red	The red color.	
Ī		green	The green color.	
Ī		blue	The blue color.	

7.14 AbstractPainterRGB888 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/AbstractPainterRGB888.hpp>
```

Public Member Functions

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

- virtual bool renderInit ()
 - Initialize rendering of a single scan line of pixels for the render.
- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0
 Get the color of the next pixel in the scan line.
- virtual void renderPixel (uint16_t *p, uint8_t red, uint8_t green, uint8_t blue)
 Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.14.1 Detailed Description

The AbstractPainterRGB888 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.14.2 Member Function Documentation

7.14.2.1 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterRGB888, PainterRGB888Bitmap, and PainterRGB888L8Bitmap.

7.14.2.2 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterRGB888Bitmap, and PainterRGB888L8Bitmap.

7.14.2.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterRGB888, PainterRGB888Bitmap, and PainterRGB888L8Bitmap.

7.14.2.4 renderPixel()

```
void renderPixel (
          uint16_t * p,
          uint8_t red,
          uint8_t green,
          uint8_t blue ) [protected], [virtual]
```

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 5,6, 5.

Parameters

ĺ	in	р	pointer into the frame buffer where the given color should be written.	
ſ		red	The red color.	
		green	The green color.	
Ī		blue	The blue color.	

7.15 AbstractPainterRGBA2222 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/AbstractPainterRGBA2222.hpp>

Public Member Functions

- virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

 Paint a designated part of the RenderingBuffer.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t newpix, uint8_t bufpix, uint8_t alpha)
 Mix colors.
- FORCE_INLINE_FUNCTION uint8_t mixColors (uint8_t R, uint8_t G, uint8_t B, uint8_t bufpix, uint8_t alpha) Mix colors.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

- virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)=0

 Get the color of the next pixel in the scan line.
- virtual void renderPixel (uint8_t *p, uint8_t red, uint8_t green, uint8_t blue)

 Renders the pixel.

Protected Attributes

· int currentX

Current x coordinate relative to the widget.

· int currentY

Current y coordinate relative to the widget.

Additional Inherited Members

7.15.1 Detailed Description

The AbstractPainterRGBA2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.15.2 Member Function Documentation

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

newpix	The newpix value.
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

uint8_t alpha) [inline]

Mix colors from a new pixel and a buffer pixel with the given alpha applied to the new pixel.

Parameters

R	The red color (0-255).
G	The green color (0-255).
В	The blue color (0-255).
bufpix	The bufpix value.
alpha	The alpha of the newpix.

Returns

The new color to write to the frame buffer.

7.15.2.3 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	X	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).

Parameters

	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Implements AbstractPainter.

Reimplemented in PainterRGBA2222, and PainterRGBA2222Bitmap.

7.15.2.4 renderInit()

```
bool renderInit ( ) [inline], [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented in PainterRGBA2222Bitmap.

7.15.2.5 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implemented in PainterRGBA2222, and PainterRGBA2222Bitmap.

7.15.2.6 renderPixel()

Renders the pixel into the frame buffer. The colors are reduced from 8,8,8 to 2,2,2.

Parameters

in	р	pointer into the frame buffer where the given color should be written.	
	red	The red color.	
	green	The green color.	
	blue	The blue color.	

7.16 AbstractPartition Class Reference

This type defines an abstract interface to a storage partition for allocating memory slots of equal size.

#include <touchgfx/common/AbstractPartition.hpp>

Public Member Functions

virtual ∼AbstractPartition ()

Virtual destructor.

virtual void * allocate (uint16_t size)

Gets the address of the next available storage slot.

virtual void * allocateAt (uint16_t index, uint16_t size)

Gets the address of the specified index.

virtual uint16_t getAllocationCount () const

Gets allocation count.

virtual uint16_t indexOf (const void *address)

Determines index of previously allocated location.

• virtual void clear ()

Prepares the Partition for new allocations.

• virtual uint16_t capacity () const =0

Gets the capacity, i.e. the maximum allocation count.

 $\bullet \ \ \text{template}{<} \text{typename T} >$

```
void * allocate ()
```

Gets the address of the next available storage slot.

• template<typename T >

```
void * allocateAt (uint16_t index)
```

Gets the address of the specified storage slot.

• template<typename T >

T & at (const uint16_t index)

Gets the object at the specified index.

• template<typename T >

const T & at (const uint16_t index) const

const version of at().

template<class T >

```
Pair < T *, uint16_t > find (const void *pT)
```

Determines if the specified object could have been previously allocated in the partition.

• void dec ()

Decreases number of allocations.

• virtual uint32_t element_size ()=0

Access to concrete element-size. Used internally.

Protected Member Functions

virtual void * element (uint16_t index)=0

Access to stored element. Used internally.

virtual const void * element (uint16 t index) const =0

Access to stored element, const version.

• AbstractPartition ()

Default constructor.

7.16.1 Detailed Description

This type defines an abstract interface to a storage partition for allocating memory slots of equal size. The "partition" is not aware of the actual types stored in the partition memory, hence it provides no mechanism for deleting C++ objects when clear()'ed.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 ~AbstractPartition()

```
~AbstractPartition ( ) [virtual]
```

Virtual destructor.

7.16.2.2 AbstractPartition()

```
AbstractPartition ( ) [protected]
```

Default constructor.

7.16.3 Member Function Documentation

Gets the address of the next available storage slot. The slot size is compared with the specified size.

Note

Asserts if 'size' is too large, or the storage is depleted.

Parameters

size	The size.

Returns

The address of an empty storage slot which contains minimum 'size' bytes.

```
7.16.3.2 allocate() [2/2]
template< typename T > void * allocate ( ) [inline]
```

Gets the address of the next available storage slot. The slot size is determined from the size of type T.

Note

Asserts if T is too large, or the storage is depleted.

Template Parameters

T Generic type paramete	r.
-------------------------	----

Returns

The address of an empty storage slot.

Gets the address of the specified index.

Note

Asserts if 'size' is too large.

Parameters

index	Zero-based index of the.
size	The size.

Returns

The address of the appropriate storage slot which contains minimum 'size' bytes.

Gets the address of the specified storage slot. The slot size is determined from the size of type T.

Note

Asserts if T is too large.

Template Parameters

T Generic type parameter.

Parameters

index Zero-based index of the.

Returns

The address of the appropriate storage slot.

Gets the object at the specified index.

Template Parameters

T Generic type parameter.

Parameters

index The index into the Partition storage where the returned object is located.

Returns

A typed reference to the object at the specified index.

Template Parameters

const version of at().

T Generic type parameter.

Parameters

index Zero-based index of the.

Returns

AT&

```
7.16.3.7 capacity()
```

```
uint16_t capacity ( ) const [pure virtual]
```

Gets the capacity, i.e. the maximum allocation count.

Returns

The maximum allocation count.

Implemented in Partition < ListOfTypes, NUMBER_OF_ELEMENTS >.

```
7.16.3.8 clear()
```

```
void clear ( ) [virtual]
```

Prepares the Partition for new allocations. Any objects present in the Partition shall not be used after invoking this method.

```
7.16.3.9 dec()
```

```
void dec ( ) [inline]
```

Decreases number of allocations.

```
7.16.3.10 element() [1/2]
```

Access to stored element. Used internally.

Parameters

index	Zero-based index of the.
-------	--------------------------

Returns

null if it fails, else a void*.

Implemented in Partition < ListOfTypes, NUMBER_OF_ELEMENTS >.

```
7.16.3.11 element() [2/2]
```

Access to stored element, const version.

Parameters

index Zero-based index of the.	index	Zero-based in	dex of the.
--------------------------------	-------	---------------	-------------

Returns

null if it fails, else a void*.

Implemented in Partition < ListOfTypes, NUMBER_OF_ELEMENTS >.

7.16.3.12 element_size()

```
uint32_t element_size ( ) [pure virtual]
```

Access to concrete element-size. Used internally.

Returns

An uint32 t.

Implemented in Partition < ListOfTypes, NUMBER_OF_ELEMENTS >.

7.16.3.13 find()

```
template< class T > Pair< T *, uint16_t > find ( const void * pT ) [inline]
```

Determines if the specified object could have been previously allocated in the partition. Since the Partition concept is loosely typed this method shall be used with care. The method does not guarantee that the found object at the returned index is a valid object. It only tests whether or not the object is within the bounds of the current partition allocations.

Template Parameters

T Generic type parameter.

Parameters

pT Pointer to the object to lookup.

Returns

If the object seems to be allocated in the Partition, a Pair object containing a typed pointer to the object and an index into the Partition storage is returned. Otherwise, a Pair < 0, 0 > is returned.

7.16.3.14 getAllocationCount()

```
uint16_t getAllocationCount ( ) const [virtual]
```

Gets allocation count.

Returns

The currently allocated storage slots.

7.16.3.15 indexOf()

Determines index of previously allocated location. Since the Partition concept is loosely typed this method shall be used with care. The method does not guarantee that the found object at the returned index is a valid object. It only tests whether or not the object is within the bounds of the current partition allocations.

Parameters

address	The location address to lookup.
---------	---------------------------------

Returns

An uint16 t.

7.17 AbstractProgressIndicator Class Reference

An abstract progress indicator.

#include <touchgfx/containers/progress_indicators/AbstractProgressIndicator.←
hpp>

Public Member Functions

AbstractProgressIndicator ()

Default constructor.

virtual ~AbstractProgressIndicator ()

Destructor.

virtual void setBackground (const Bitmap &bmpBackground)

Sets the background image.

• virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the position and dimension of the actual progress indicator.

virtual int16_t getProgressIndicatorX () const

Gets progress indicator x coordinate.

virtual int16_t getProgressIndicatorY () const

Gets progress indicator y coordinate.

virtual int16_t getProgressIndicatorWidth () const

Gets progress indicator width.

virtual int16 t getProgressIndicatorHeight () const

Gets progress indicator height.

• virtual void setRange (int16_t min, int16_t max, uint16_t steps=0, uint16_t minStep=0)

Sets the range for the progress indicator.

- virtual void getRange (int16_t &min, int16_t &max, uint16_t &steps, uint16_t &minStep) const
- virtual void getRange (int16_t &min, int16_t &max, uint16_t &steps) const

Gets the range.

virtual void getRange (int16_t &min, int16_t &max) const

Gets the range.

• virtual void setValue (int value)

Sets a value.

• virtual int getValue () const

Gets the value.

virtual uint16_t getProgress (uint16_t range=100) const

Gets the progress.

Protected Attributes

· Image background

The background image.

· Container progressIndicatorContainer

The container that holds the actual progress indicator.

• int16_t rangeMin

The range minimum.

int16_t rangeMax

The range maximum.

• uint16_t currentValue

The current value.

uint16_t rangeSteps

The range steps.

uint16_t rangeStepsMin

The range steps minimum.

Additional Inherited Members

7.17.1 Detailed Description

The AbstractProgressIndicator declares methods that provides the basic mechanisms and tools to implement a progress indicator. For more specific implementations see classes that inherit from AbstractProgressIndicator.

See also

BoxProgess CircleProgress

ImageProgress

LineProgress

TextProgress

7.17.2 Constructor & Destructor Documentation

7.17.2.1 AbstractProgressIndicator()

AbstractProgressIndicator ()

Initialized the progress indicator with default range 0-100.

```
See also
```

setRange

```
7.17.2.2 \simAbstractProgressIndicator()
```

```
~AbstractProgressIndicator ( ) [virtual]
```

Destructor.

7.17.3 Member Function Documentation

7.17.3.1 getProgress()

Gets the current progress based on the range set by setRange() and the value set by setValue().

Parameters

range The range.	
------------------	--

Returns

The progress.

See also

setRange, setValue

7.17.3.2 getProgressIndicatorHeight()

```
int16_t getProgressIndicatorHeight ( ) const [virtual]
```

Gets progress indicator height.

Returns

The progress indicator height.

7.17.3.3 getProgressIndicatorWidth()

```
int16_t getProgressIndicatorWidth ( ) const [virtual]
```

Gets progress indicator width.

Returns

The progress indicator width.

7.17.3.4 getProgressIndicatorX()

```
int16_t getProgressIndicatorX ( ) const [virtual]
```

Gets progress indicator x coordinate.

Returns

The progress indicator x coordinate.

7.17.3.5 getProgressIndicatorY()

```
int16_t getProgressIndicatorY ( ) const [virtual]
```

Gets progress indicator y coordinate.

Returns

The progress indicator y coordinate.

7.17.3.6 getRange() [1/3]

Gets the range set by setRange().

Parameters

out	min	The minimum input value.
out	max	The maximum input value.
out	steps	The steps in which to report progress.
out	minStep	The step which the minimum input value is mapped to.

7.17.3.7 getRange() [2/3]

Gets the range set by setRange().

Parameters

	out	min	The minimum input value.
ĺ	out	max	The maximum input value.
	out	steps	The steps in which to report progress.

Gets the range set by setRange().

Parameters

out	min	The minimum input value.
out	max	The maximum input value.

7.17.3.9 getValue()

```
int getValue ( ) const [virtual]
```

Gets the current value set by setValue().

Returns

The value.

7.17.3.10 setBackground()

```
void setBackground ( {\tt const\ Bitmap\ \&\ bmpBackground\ )} \quad [{\tt virtual}]
```

Sets the background image. The width and height of the widget is updated according to the dimension of the image.

Parameters

bmpBackground	The background bitmap.

7.17.3.11 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the actual progress indicator relative to the background image.

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the box progress indicator.
height	The height of the box progress indicator.

Reimplemented in LineProgress, CircleProgress, BoxProgress, ImageProgress, and TextProgress.

7.17.3.12 setRange()

```
void setRange (
    int16_t min,
    int16_t max,
    uint16_t steps = 0,
    uint16_t minStep = 0 ) [virtual]
```

Sets the range for the progress indicator. The range is the values that are given to the progress indicator while progressing through the task at hand. If an app needs to work through 237 items to finish a task, the range should be set to (0, 237) assuming that 0 items is the minimum. Though the minimum is often 0, it is possible to customize this.

The steps parameter, is used to specify at what granularity you want the progress indicator to report a new progress value. If the 237 items to be reported as 0%, 10%, 20%, ... 100%, the steps should be set to 10 as there are ten steps from 0% to 100%. If you want to update a widget which is 150 pixels wide, you might want to set steps to 150 to get a new progress value for every pixel. If you are updating a clock and want this to resemble an analog clock, you might want to use 12 or perhaps 60 as number of steps.

The minStep parameter is used when the value min should give a progress different from 0. For example, if progress is a clock face, you want to count from 0..1000 and you want progress per minute, but want to make sure that 0 is not a blank clock face, but instead you want 1 minute to show, the setRange(0, 1000, 60, 1) will make sure that as values progress from 0 to 1000, getProgress() start from 1 and goes up to 60. Another example could be a BoxProgress with a TextProgress on top and you want to make sure that "0%" will always show in the box, use something like setRange(0, 1000, 200, 40) if your box is 200 wide and "0%" is 40 wide.

Parameters

min	The minimum input value.
max	The maximum input value.
steps	The steps in which to report progress.
minStep	The step which the minimum input value is mapped to.

See also

setValue, getProgress

7.17.3.13 setValue()

```
void setValue (
int value ) [virtual]
```

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

Reimplemented in CircleProgress, LineProgress, ImageProgress, TextProgress, and BoxProgress.

7.18 AbstractShape Class Reference

Simple widget capable of drawing a abstractShape.

```
#include <touchgfx/widgets/canvas/AbstractShape.hpp>
```

Classes

struct ShapePoint

Defines an alias representing the array of points making up the abstract shape.

Public Member Functions

· AbstractShape ()

Constructs a new AbstractShape.

virtual ~AbstractShape ()

Virtual Destructor.

• virtual int getNumPoints () const =0

Gets number points used to make up the shape.

virtual void setCorner (int i, CWRUtil::Q5 x, CWRUtil::Q5 y)=0

Sets a corner of the shape.

virtual CWRUtil::Q5 getCornerX (int i) const =0

Gets the x coordinate of a corner.

virtual CWRUtil::Q5 getCornerY (int i) const =0

Gets the y coordinate of a corner.

```
• template<typename T >
```

```
void setShape (ShapePoint< T > *points)
```

Sets a shape the struct Points.

• template<typename T >

```
void setShape (const ShapePoint< T > *points)
```

Sets a shape the struct Points.

```
• template<typename T >
```

```
void setOrigin (T x, T y)
```

Sets the position of (0,0).

 $\bullet \ \ template {<} typename \ T >$

```
void moveOrigin (T x, T y)
```

Moves the start point for this AbstractShape.

template<typename T >

```
void getOrigin (T &dx, T &dy) const
```

Gets the start coordinates for the line.

template<typename T >

void setAngle (T angle)

Sets the angle to turn the abstractShape.

• template<typename T >

void getAngle (T &angle)

Gets the abstractShape's angle.

 $\bullet \;\; template {<} typename \; T >$

void updateAngle (T angle)

Sets the angle to turn the abstractShape.

• int getAngle () const

Gets the current angle of the abstractShape.

• template<typename T >

void setScale (T newXScale, T newYScale)

Scale the AbstractShape.

• template<typename T >

void setScale (T scale)

Scale the AbstractShape.

• template<typename T >

void updateScale (T newXScale, T newYScale)

Scale the AbstractShape.

• template<typename T >

void getScale (T &x, T &y) const

Gets the x scale and y scale.

virtual bool drawCanvasWidget (const Rect &invalidatedArea) const

Draws the AbstractShape.

void updateAbstractShapeCache ()

Updates the abstractShape cache.

Protected Member Functions

virtual void setCache (int i, CWRUtil::Q5 x, CWRUtil::Q5 y)=0

Sets the cached coordinates of a given corner.

virtual CWRUtil::Q5 getCacheX (int i) const =0

Gets cached x coordinate of a corner.

• virtual CWRUtil::Q5 getCacheY (int i) const =0

Gets cached y coordinate of a corner.

virtual Rect getMinimalRect () const

Gets minimal rectangle containing the abstractShape.

Additional Inherited Members

7.18.1 Detailed Description

Simple widget capable of drawing a abstractShape. The abstractShape can be scaled and rotated around 0,0. Note that the y axis goes down, so a abstractShape that goes up must be given negative coordinates.

See also

CanvasWidget

tparam T The type of the points used for the abstractShape. Must be int or float.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 AbstractShape()

```
AbstractShape ( )
```

Constructs a new AbstractShape.

7.18.2.2 \sim AbstractShape()

```
~AbstractShape ( ) [virtual]
```

Virtual Destructor.

7.18.3 Member Function Documentation

7.18.3.1 drawCanvasWidget()

Draws the AbstractShape. This class supports partial drawing, so only the area described by the rectangle will be drawn.

Parameters

	invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.	
--	-----------------	--	--

Returns

true if it succeeds, false if it fails.

Implements CanvasWidget.

7.18.3.2 getAngle() [1/2]

```
template< typename T > void getAngle ( T & angle ) [inline]
```

Gets the abstractShape's angle.

Template Parameters

T Generic type parameter.

Parameters

out	angle	The current abstractShape angle relative to 0.
-----	-------	--

```
7.18.3.3 getAngle() [2/2] template< typename T > T getAngle ( ) const [inline] Gets the current angle of the abstractShape.
```

Returns

The angle of the AbstractShape.

7.18.3.4 getCacheX()

Gets cached x coordinate of a corner.

Parameters

```
i Zero-based index of the corner.
```

Returns

The cached x coordinate.

Implemented in Shape< POINTS >.

7.18.3.5 getCacheY()

Gets cached y coordinate of a corner.

Parameters

```
i Zero-based index of the corner.
```

Returns

The cached y coordinate.

Implemented in Shape < POINTS >.

7.18.3.6 getCornerX()

Gets the x coordinate of a corner.

Parameters

```
i Zero-based index of the corner.
```

Returns

The corner x coordinate.

Implemented in Shape < POINTS >.

7.18.3.7 getCornerY()

Gets the y coordinate of a corner.

Parameters

```
i Zero-based index of the corner.
```

Returns

The corner y coordinate.

Implemented in Shape < POINTS >.

7.18.3.8 getMinimalRect()

```
Rect getMinimalRect ( ) const [protected], [virtual]
```

Gets minimal rectangle containing the abstractShape. Used for invalidating only the required part of the screen.

Returns

The minimal rectangle.

Reimplemented from CanvasWidget.

7.18.3.9 getNumPoints()

```
int getNumPoints ( ) const [pure virtual]
```

Gets number points used to make up the shape.

Returns

The number points.

Implemented in Shape < POINTS >.

7.18.3.10 getOrigin()

```
template< typename T > void getOrigin (  \mbox{T \& } dx,   \mbox{T \& } dy \mbox{ ) const [inline]}
```

Gets the start coordinates for the line.

Template Parameters

T Generic type parameter, either int or flo	at.
---	-----

Parameters

out	dx	The x coordinate.
out	dy	The y coordinate.

7.18.3.11 getScale()

```
template< typename T > void getScale ( T & x, T & y ) const [inline]
```

Gets the x scale and y scale of the shape as previously set using setScale. Default is 1 for both x scale and y scale.

Template Parameters

T	Generic type parameter, either int or float.

Parameters

out	X	Scaling of x coordinates.
out	У	Scaling of y coordinates.

See also

setScale

7.18.3.12 moveOrigin()

Moves the start point for this AbstractShape. The rectangle that surrounds the old and new area covered by the shape will be invalidated.

Template Parameters

T Generic type parameter, either int or float.

Parameters

	The x coordinate of the shapes position $(0,0)$.
У	The y coordinate of the shapes position $(0,0)$.

Note

The area containing the AbstractShape is invalidated before and after the change.

See also

setOrigin()

7.18.3.13 setAngle()

Sets the angle to turn the abstractShape. 0 degrees is straight up and 90 degrees is 3 o'clock.

Template Parameters

T	Generic type parameter.
---	-------------------------

Parameters

angle	The angle to turn the abstractShape to relative to 0 (straight up), not relative to the previous angle.
-------	---

Note

The area containing the AbstractShape is not invalidated.

See also

updateAngle()

7.18.3.14 setCache()

```
void setCache (
          int i,
          CWRUtil::Q5 x,
          CWRUtil::Q5 y ) [protected], [pure virtual]
```

Sets the cached coordinates of a given corner. The coordinates in the cache are the coordinates from the corners after rotating and scaling the coordinate.

Parameters

i	Zero-based index of the corner.	
Х	The x coordinate.	
У	The y coordinate.	

Implemented in Shape < POINTS >.

7.18.3.15 setCorner()

```
void setCorner (
    int i,
    CWRUtil::Q5 x,
    CWRUtil::Q5 y) [pure virtual]
```

Sets a corner of the shape in Q5 format.

Parameters

i	Zero-based index of the corner.
Х	The x coordinate in Q5 format.
У	The y coordinate in Q5 format.

Note

Remember to call updateAbstractShapeCache() to make sure that the cached outline of the shape is correct.

See also

updateAbstractShapeCache

Implemented in Shape < POINTS >.

7.18.3.16 setOrigin()

```
template< typename T > void setOrigin ( T x, T y ) [inline]
```

Sets the position of (0,0) used when the abstractShape was created. This means that all coordinates initially used when created the shape are moved relative to these given offsets. Calling setOrigin() again, will not add to the previous settings of setOrigin() but will replace the old values for origin.

Template Parameters

Parameters

X	The x coordinate of the shapes position $(0,0)$.
У	The y coordinate of the shapes position (0,0).

Note

The area containing the AbstractShape is not invalidated.

See also

moveOrigin()

Scale the AbstractShape the given amounts in the x direction and the y direction.

Template Parameters

T Generic type parameter, either int or float.

Parameters

newXScale	The new scale in the x direction.
newYScale	The new scale in the y direction.

Note

The area containing the AbstractShape is not invalidated.

See also

getScale, updateScale

Scale the AbstractShape the given amount in the x direction and the y direction.

Template Parameters

T Generic type parameter, either int or float.

Parameters

scale	The scale in the x direction.
-------	-------------------------------

Note

The area containing the AbstractShape is not invalidated.

See also

getScale

Sets a shape the struct Points. The cached outline of the shape is automatically updated.

Template Parameters

T Generic type parameter, either int or float.

Parameters

	in	points	The points that make up the shape.]
--	----	--------	------------------------------------	---

Note

The area containing the shape is not invalidated.

Sets a shape the struct Points. The cached outline of the shape is automatically updated.

Template Parameters

T Generic type parameter, either int or float.

Parameters

ir	points	The points that make up the shape.
----	--------	------------------------------------

Note

The area containing the shape is not invalidated.

7.18.3.21 updateAbstractShapeCache()

```
void updateAbstractShapeCache ( )
```

Updates the abstractShape cache. The cache is used to be able to quickly redraw the AbstractShape without calculating the points that make up the abstractShape (with regards to scaling and rotation).

7.18.3.22 updateAngle()

```
template< typename T > void updateAngle ( T angle ) [inline]
```

Sets the angle to turn the abstractShape. 0 degrees is straight up and 90 degrees is 3 o'clock.

Template Parameters

```
T Generic type parameter.
```

Parameters

angle The angle to turn the abstract	tShape.
--------------------------------------	---------

Note

The area containing the AbstractShape is invalidated before and after the change.

See also

setAngle()

7.18.3.23 updateScale()

Scale the AbstractShape the given amounts in the x direction and the y direction.

Template Parameters

T	Generic type parameter, either int or float.
---	--

Parameters

newXScale	The new scale in the x direction.
newYScale	The new scale in the y direction.

Note

The area containing the AbstractShape is invalidated before and after the change.

See also

setScale()

7.19 AnalogClock Class Reference

An analog clock.

#include <touchgfx/containers/clock/AnalogClock.hpp>

Public Member Functions

AnalogClock ()

Default constructor.

virtual ∼AnalogClock ()

Destructor.

· virtual void setBackground (const BitmapId backgroundBitmapId)

Sets the background image of the clock.

virtual void setBackground (const BitmapId backgroundBitmapId, int16_t rotationCenterX, int16_t rotation
 — CenterY)

Sets the background image of the clock and the rotation center of the clock.

virtual void setRotationCenter (int16_t rotationCenterX, int16_t rotationCenterY)

Sets the rotation center of the clock.

virtual void setupHourHand (const BitmapId hourHandBitmapId, int16_t rotationCenterX, int16_t rotation
 — CenterY)

Sets up the hour hand.

 virtual void setupMinuteHand (const BitmapId minuteHandBitmapId, int16_t rotationCenterX, int16_← t rotationCenterY)

Sets up the minute hand.

virtual void setupSecondHand (const BitmapId secondHandBitmapId, int16_t rotationCenterX, int16_
t rotationCenterY)

Sets up the second hand.

virtual void setHourHandMinuteCorrection (bool active)

Sets hour hand minute correction.

· virtual bool getHourHandMinuteCorrection () const

Gets hour hand minute correction.

virtual void setMinuteHandSecondCorrection (bool active)

Sets minute hand second correction.

· virtual bool getMinuteHandSecondCorrection () const

Gets minute hand second correction.

virtual void setAnimation (uint16_t duration=10, EasingEquation animationProgressionEquation=Easing←
 Equations::backEaseInOut)

Setup the clock to use animation for hand movements.

virtual uint16_t getAnimationDuration ()

Gets the animation duration.

• virtual void initializeTime24Hour (uint8 t hour, uint8 t minute, uint8 t second)

Sets the time with input format as 24H. No animations are performed.

• virtual void initializeTime12Hour (uint8_t hour, uint8_t minute, uint8_t second, bool am)

Sets the time with input format as 12H. No animations are performed.

Protected Member Functions

virtual void updateClock ()

Updates the visual representation of the clock.

 virtual void setupHand (TextureMapper &hand, const BitmapId bitmapId, int16_t rotationCenterX, int16_← t rotationCenterY)

Sets up a given the hand.

- virtual float convertHandValueToAngle (uint8_t steps, uint8_t handValue, uint8_t secondHandValue=0) const Convert hand value to angle.
- virtual bool animationEnabled () const

Is animation enabled.

Protected Attributes

· Image background

The background image of the AnalogClock.

AnimationTextureMapper hourHand

TextureMapper that represents the hourHand.

AnimationTextureMapper minuteHand

TextureMapper that represents the minuteHand.

AnimationTextureMapper secondHand

TextureMapper that represents the secondHand.

· EasingEquation animationEquation

The easing equation used by hand animations.

· uint16 t animationDuration

The duration of hand animations. If 0 animations are disabled.

int16 t clockRotationCenterX

The rotation point (X) of the hands.

· int16 t clockRotationCenterY

The rotation point (Y) of the hands.

· uint8 t lastHour

The last know hour value.

· uint8 t lastMinute

The last know minute value.

· uint8 t lastSecond

The last know second value.

bool hourHandMinuteCorrectionActive

Is hour hand minute correction active.

· bool minuteHandSecondCorrectionActive

Is minute hand second correction active.

Additional Inherited Members

7.19.1 Detailed Description

An analog clock. Should be supplied with images for the background, hour hand, minute hand and the optional second hand. You setup the AnalogClock by specifying the rotation point of each hand as well as the global rotation point of the clock.

You can customize the behavior of the AnalogClock in respect to animations and relations between the hands e.g. the hour hand moves gradually towards the next hour as the minute hand progresses (setHourHandMinute Correction())

7.19.2 Member Function Documentation

7.19.2.1 animationEnabled()

```
bool animationEnabled ( ) const [protected], [virtual]
```

Is animation enabled.

Returns

true if animation is enabled.

7.19.2.2 convertHandValueToAngle()

Convert hand value to angle.

Parameters

steps	Number of steps the primary hand value is divided into (e.g. 60 for minutes/seconds and 12 for hour).
handValue	The actual value for the hand in question (in the range [0;steps]).
secondHandValue	(Optional) If the angle should be corrected for a secondary hand its value should be specified here (in the range [0;60]). This is the case when setHourHandMinuteCorrection(true) or setMinuteHandSecondCorrection(true) is selected.

Returns

The converted value to angle.

7.19.2.3 getAnimationDuration()

```
uint16_t getAnimationDuration ( ) [inline], [virtual]
```

Returns

The animation duration.

7.19.2.4 getHourHandMinuteCorrection()

```
bool getHourHandMinuteCorrection ( ) const [virtual]
```

Gets hour hand minute correction.

Returns

true if hour hand minute correction is active.

See also

setHourHandMinuteCorrection

7.19.2.5 getMinuteHandSecondCorrection()

```
bool getMinuteHandSecondCorrection ( ) const [virtual]
```

Gets minute hand second correction.

Returns

true if minute hand second correction is active.

See also

setHourHandMinuteCorrection

7.19.2.6 initializeTime12Hour()

Sets the time with input format as 12H. No animations are performed regardless of the animation settings. This is often useful when setting up the AnalogClock where you do not want an initial animation. Note that this does not affect any selected presentation formats.

Parameters

hour	The hours (in 12H format).
minute	The minutes (in 12H format).
second	The seconds (in 12H format).
am	AM/PM setting. True = AM, false = PM.

7.19.2.7 initializeTime24Hour()

Sets the time with input format as 24H. No animations are performed regardless of the animation settings. This is often useful when setting up the AnalogClock where you do not want an initial animation. Note that this does not affect any selected presentation formats.

Parameters

hour	The hours (in 24H format).
minute	The minutes (in 24H format).
second	The seconds (in 24H format).

7.19.2.8 setAnimation()

[virtual]

Parameters

duration	(Optional) The animation duration.
animationProgressionEquation	(Optional) The animation progression equation.

Sets the background image of the clock. The clock rotation center is set to the background image center. The clock rotation center is the point that the clock hands rotates around.

Parameters

background←	Identifier for the background bitmap.
BitmapId	

7.19.2.10 setBackground() [2/2]

Sets the background image of the clock and the rotation center of the clock. The clock rotation center is the point that the clock hands rotates around.

Parameters

background⊷ BitmapId	Identifier for the background bitmap.
rotationCenterX	The rotation center x coordinate.
rotationCenterY	The rotation center y coordinate.

7.19.2.11 setHourHandMinuteCorrection()

If set to true the hour hand will be positioned between the current hour and the next depending on the minute hands position.

Parameters

active	true to use hour hand correction.

See also

getHourHandMinuteCorrection

7.19.2.12 setMinuteHandSecondCorrection()

```
\begin{tabular}{ll} {\tt void setMinuteHandSecondCorrection (} \\ {\tt bool } \begin{tabular}{ll} {\tt active ()} & [{\tt virtual}] \end{tabular}
```

If set to true the minute hand will be positioned between the current minute and the next depending on the second hands position.

Parameters

active	true to use.
active	true to use.

See also

setMinuteHandSecondCorrection

7.19.2.13 setRotationCenter()

Sets the rotation center of the clock. The clock rotation center is the point that the clock hands rotates around.

Parameters

rotationCenterX	The rotation center x coordinate.
rotationCenterY	The rotation center y coordinate.

7.19.2.14 setupHand()

Sets up a given the hand.

Parameters

in,out	hand	Reference to the hand being setup.
	bitmapld	The bitmap identifier for the given hand.
	rotationCenterX	The hand rotation center x coordinate.
	rotationCenterY	The hand rotation center y coordinate.

7.19.2.15 setupHourHand()

Sets up the hour hand. The specified rotation center is the point of the hand that is to be placed on top of the clock rotation center. That is the point that the hand rotates around. The rotation point is relative to the supplied bitmap but can be placed outside it.

If not called the hour hand will just be omitted.

Parameters

hourHand⊷ BitmapId	Identifier for the hour hand bitmap.
rotationCenterX	The hand rotation center x coordinate.
rotationCenterY	The hand rotation center y coordinate.

7.19.2.16 setupMinuteHand()

Sets up the minute hand. The specified rotation center is the point of the hand that is to be placed on top of the clock rotation center. That is the point that the hand rotates around. The rotation point is relative to the supplied bitmap but can be placed outside it.

If not called the minute hand will just be omitted.

Parameters

minuteHand↔ BitmapId	Identifier for the minute hand bitmap.
rotationCenterX	The hand rotation center x coordinate.
rotationCenterY	The hand rotation center y coordinate.

7.19.2.17 setupSecondHand()

Sets up the second hand. The specified rotation center is the point of the hand that is to be placed on top of the clock rotation center. That is the point that the hand rotates around. The rotation point is relative to the supplied bitmap but can be placed outside it.

If not called the second hand will just be omitted.

Parameters

secondHand↔ BitmapId	Identifier for the second hand bitmap.
rotationCenterX	The hand rotation center x coordinate.
rotationCenterY	The hand rotation center y coordinate.

7.19.2.18 updateClock()

```
virtual void updateClock ( ) [protected], [virtual]
```

Updates the visual representation of the clock.

Implements AbstractClock.

7.20 AnimatedImage Class Reference

A widget capable of basic animation using a range of bitmaps.

```
#include <touchgfx/widgets/AnimatedImage.hpp>
```

Public Member Functions

AnimatedImage (const BitmapId &start, const BitmapId &end, const uint8_t &updateInterval=1)

Constructs an AnimatedImage.

AnimatedImage (const uint8_t &updateInterval=1)

Constructor.

virtual void startAnimation (const bool &rev, const bool &reset=false, const bool &loop=false)

Starts the animation.

• virtual void stopAnimation ()

Stops and resets the animation.

virtual void pauseAnimation ()

Toggles the running state of an animation.

virtual void handleTickEvent ()

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

void setDoneAction (GenericCallback< const AnimatedImage & > &callback)

Associates an action to be performed when the animation of the AnimatedImage is done.

• bool isRunning ()

Gets the running state of the AnimatedImage.

• bool isAnimatedImageRunning ()

Gets the running state of the AnimatedImage.

bool isReverse ()

Query if this object is running in reverse.

• void setBitmaps (Bitmapld start, Bitmapld end)

Sets the bitmaps that are used by the animation.

void setUpdateTicksInterval (uint8_t updateInterval)

Sets the update interval.

virtual uint16_t getType () const

For GUI testing only.

Protected Member Functions

virtual void setBitmap (const Bitmap &bmp)

Is not public available. Use setBitmaps instead.

Protected Attributes

GenericCallback< const AnimatedImage & > * animationDoneAction

Pointer to the callback being executed when animation is done.

· Bitmapld startld

Id of first bitmap in animation.

· Bitmapld endld

Id of second bitmap in animation.S.

• uint8_t updateTicksInterval

Number of ticks required between each animation update (image change).

uint8_t ticksSinceUpdate

Number of ticks since last animation update.

· bool reverse

If true, run in reverse direction (last to first).

bool loopAnimation

If true, continuously loop animation.

· bool running

If true, animation is running.

Additional Inherited Members

7.20.1 Detailed Description

The AnimatedImage is capable of running the animation from start to end or in reverse order, end to start. It is capable doing a single animation or looping the animation until stopped or paused. See animation_example far a demonstration of how to use of this widget.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 AnimatedImage() [1/2]

The start and the end specifies the range of bitmaps to be used for animation. The update interval defines how often the animation should be updated. The animation will iterate over the bitmaps that lies between the IDs of start and end, both included.

Parameters

start	Defines the start of the range of images in the animation.
end	Defines the end of the range of images in the animation.
updateInterval	Defines the number of ticks between each animation step. Higher value results in a slower animation.

7.20.2.2 AnimatedImage() [2/2]

Constructs an AnimatedImage without initializing bitmaps.

Note

The bitmaps to display must be configured through set setBitmaps function before this widget displays anything.

Parameters

updateInterval	Defines the number of ticks between each animation step. Higher value results in a slower
	animation.

7.20.3 Member Function Documentation

```
7.20.3.1 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_ANIMATEDIMAGE.

Reimplemented from Image.

7.20.3.2 handleTickEvent()

```
virtual void handleTickEvent ( ) [virtual]
```

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

See also

Application::registerTimerWidget

Reimplemented from Drawable.

7.20.3.3 isAnimatedImageRunning()

```
bool isAnimatedImageRunning ( ) [inline]
```

Gets the running state of the AnimatedImage.

Returns

true if the animation is currently running, false otherwise.

7.20.3.4 isReverse()

```
bool isReverse ( ) [inline]
```

Query if this object is running in reverse.

Returns

true if the animation is performed in reverse order.

7.20.3.5 isRunning()

```
bool isRunning ( ) [inline]
```

Gets the running state of the AnimatedImage.

Returns

true if the animation is currently running, false otherwise.

7.20.3.6 pauseAnimation()

```
void pauseAnimation ( ) [virtual]
```

Toggles the running state of an animation. Pauses the animation if the animation is running. Continues the animation if previously paused.

7.20.3.7 setBitmap()

Is not public available. Use setBitmaps instead. Internally in AnimatedImage use Image::setBitmap(...).

Parameters

bmp The bitmap.	
-----------------	--

Reimplemented from Image.

7.20.3.8 setBitmaps()

Sets the bitmaps that are used by the animation.

The animation will iterate over the bitmaps that lies between the IDs of start and end, both included.

Parameters

start	Defines the start of the range of images in the animation.
end	Defines the end of the range of images in the animation.

7.20.3.9 setDoneAction()

Associates an action to be performed when the animation of the AnimatedImage is done.

Parameters

uted when done. The callback is given the animated image	callback The callback is executed when done.
--	--

7.20.3.10 setUpdateTicksInterval()

Sets the update interval.

Parameters

updateInterval	Defines the number of ticks between each animation step. Higher value results in a slower
	animation.

7.20.3.11 startAnimation()

Starts the animation.

Parameters

rev	Defines if the animation should be performed in reverse order.
reset	Defines if the animation should reset and start from the first (or last if reverse order) bitmap.
loop	Defines if the animation should loop or do a single animation.

7.20.3.12 stopAnimation()

```
void stopAnimation ( ) [virtual]
```

Stops and resets the animation.

7.21 AnimatedImageButtonStyle < T > Class Template Reference

An animated image button style. An animated image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show the first or last image of an animated image depending on the state of the button (pressed or released). When the state changes the button will show the sequence of images in forward or reversed order.

#include <touchgfx/containers/buttons/AnimatedImageButtonStyle.hpp>

Public Member Functions

· AnimatedImageButtonStyle ()

Default constructor.

virtual ~AnimatedImageButtonStyle ()

Destructor.

void setBitmaps (const Bitmap &bmpStart, const Bitmap &bmpEnd)

Sets the bitmaps.

void setBitmapXY (uint16_t x, uint16_t y)

Sets bitmap xy.

void setUpdateTicksInterval (uint8_t updateInterval)

Sets update ticks interval.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

• AnimatedImage buttonAnimatedImage

The button animated image.

7.21.1 Detailed Description

```
\label{template} \begin{tabular}{ll} template < class T > \\ class touchgfx:: Animated Image Button Style < T > \\ \end{tabular}
```

The AnimatedImageButtonStyle will set the size of the enclosing container (normally AbstractButtonContainer) to the size of the first Bitmap. This can be overridden by calling setWidth/setHeight after setting the bitmaps.

The position of the bitmap can be adjusted with setBitmapXY (default is upper left corner).

Template Parameters

T | Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer

7.21.2 Member Function Documentation

7.21.2.1 setBitmaps()

Parameters

bmpStart	The bitmap start.
bmpEnd	The bitmap end.

7.21.2.2 setBitmapXY()

Parameters

X	An uint16_t to process.
У	An uint16_t to process.

7.21.2.3 setUpdateTicksInterval()

Parameters

updateInterval	The update interval.
----------------	----------------------

7.22 AnimationTextureMapper::AnimationSetting Struct Reference

Information about how a specific animation parameter should be animated.

#include <touchgfx/widgets/AnimationTextureMapper.hpp>

Public Attributes

· bool animationActive

Should this animation be performed.

float animationStart

The animation start value.

float animationEnd

The animation end value.

uint16_t animationDelay

A delay that is applied before animation start. Expressed in ticks.

· uint16 t animationDuration

The complete duration of the animation. Expressed in ticks.

EasingEquation animationProgressionEquation

EasingEquation expressing the development of the value during the animation.

7.23 AnimationTextureMapper Class Reference

A texture mapper with animation capabilities.

#include <touchgfx/widgets/AnimationTextureMapper.hpp>

Classes

· struct AnimationSetting

Information about how a specific animation parameter should be animated.

Public Types

enum AnimationParameter { X_ROTATION = 0, Y_ROTATION, Z_ROTATION, SCALE }

Values that represent different animation parameter.

Public Member Functions

AnimationTextureMapper ()

Default constructor.

virtual ~AnimationTextureMapper ()

Destructor.

void setTextureMapperAnimationStepAction (GenericCallback< const AnimationTextureMapper & > &callback)

Associates an action to be performed when the animation steps.

void setTextureMapperAnimationEndedAction (GenericCallback < const AnimationTextureMapper & > &callback)

Associates an action to be performed when the animation ends.

· virtual bool isTextureMapperAnimationRunning () const

Gets whether or not the animation is running.

 virtual void setupAnimation (AnimationParameter parameter, float endValue, uint16_t duration, uint16_t delay, EasingEquation progressionEquation=&EasingEquations::linearEaseNone)

Sets up the animation for a specific parameter (angle/scale) for the next animation.

• virtual void startAnimation ()

Starts the animation.

virtual void cancelAnimationTextureMapperAnimation ()

Cancel move animation.

virtual uint16_t getAnimationStep ()

Gets the current animation step.

Static Public Attributes

static const int NUMBER_OF_ANIMATION_PARAMETERS = SCALE + 1
 Number of animation parameters.

Protected Types

enum AnimationState { ANIMATION_FINISHED = 0, ANIMATION_DELAYED, ANIMATION_RUNNING }
 Values that represent different states during an animation.

Protected Member Functions

virtual void handleTickEvent ()

The tick handler that handles the actual animation steps.

Protected Attributes

AnimationSetting animations [NUMBER_OF_ANIMATION_PARAMETERS]

Descriptions of the animation of specific animation parameters.

GenericCallback < const AnimationTextureMapper &> * textureMapperAnimationStepCallback

Animation has performed a step Callback.

• GenericCallback< const AnimationTextureMapper &> * textureMapperAnimationEndedCallback

Animation ended Callback.

uint16 t animationCounter

Counter that is equal to the current step in the animation.

· bool animationRunning

Boolean that is true if the animation is running.

Additional Inherited Members

7.23.1 Detailed Description

A texture mapper with animation capabilities. Note that the angles of the TextureMapper is moved to the [0; 2PI] range at the beginning at the animation. The end angles should be relative to this and are limited to values in the range [-32.7; 32.7].

See also

TextureMapper

7.23.2 Member Enumeration Documentation

7.23.2.1 AnimationParameter

enum AnimationParameter

Enumerator

X_ROTATION	Rotation around the X axis.
Y_ROTATION	Rotation around the Y axis.
Z_ROTATION	Rotation around the Z axis.
SCALE	Scaling of the image.

7.23.2.2 AnimationState

```
enum AnimationState [protected]
```

Enumerator

ANIMATION_FINISHED	The animation is finished.
ANIMATION_DELAYED	The animation is in the delay mode.
ANIMATION_RUNNING	The animation is currently runnnig.

7.23.3 Constructor & Destructor Documentation

7.23.3.1 AnimationTextureMapper()

AnimationTextureMapper ()

Default constructor.

7.23.3.2 ~AnimationTextureMapper()

```
~AnimationTextureMapper ( ) [virtual]
```

Destructor. Destroys the AnimationTextureMapper.

7.23.4 Member Function Documentation

7.23.4.1 cancelAnimationTextureMapperAnimation()

```
void cancelAnimationTextureMapperAnimation ( ) [virtual]
```

Cancel move animation.

7.23.4.2 getAnimationStep()

```
uint16_t getAnimationStep ( ) [virtual]
```

Returns

The current animation step.

7.23.4.3 handleTickEvent()

```
void handleTickEvent ( ) [protected], [virtual]
```

The tick handler that handles the actual animation steps.

Reimplemented from Drawable.

7.23.4.4 isTextureMapperAnimationRunning()

```
bool isTextureMapperAnimationRunning ( ) const [virtual]
```

Gets whether or not the animation is running.

Returns

true if the animation is running.

7.23.4.5 setTextureMapperAnimationEndedAction()

Associates an action to be performed when the animation ends.

Parameters

callback The callback to be executed. The callback will be given a reference to the AnimationTextureMapper.

See also

GenericCallback

7.23.4.6 setTextureMapperAnimationStepAction()

Associates an action to be performed when the animation steps. Will not be called during delay period.

Parameters

callback The callback to be executed. The callback will be given a reference to the AnimationTextureMapper.

See also

GenericCallback

7.23.4.7 setupAnimation()

```
uint16_t delay,
EasingEquation progressionEquation = &EasingEquations::linearEaseNone ) [virtual]
```

Sets up the animation for a specific parameter (angle/scale) for the next animation. The specific parameter is chosen using the AnimationType enum. AnimationTypes that are not setup using this method will keep their value during the animation.

Parameters

parameter	The parameter which animation details are being specified.
endValue	The end value for the parameter.
duration	The duration for the animation of this parameter. Specified in ticks.
delay	The delay for the animation of this parameter. Specified in ticks.
progressionEquation	the progression equation for the animation of this parameter.

7.23.4.8 startAnimation()

```
void startAnimation ( ) [virtual]
```

Starts the animation from the current position to the specified end angles/scale. The progression of the angles/scale during the animation is described by the supplied EasingEquations.

7.24 Application Class Reference

The Application class is the main interface for manipulating screen contents.

```
#include <touchgfx/Application.hpp>
```

Public Member Functions

• Screen * getCurrentScreen ()

Gets the current screen.

• virtual void switchScreen (Screen *newScreen)

Switch to another Screen.

virtual void appSwitchScreen (uint8_t screenId)

An application specific function for switching screen.

virtual void draw ()

Initiate a draw operation of the entire screen.

virtual void draw (Rect &rect)

Initiate a draw operation of the specified region of the screen.

virtual void handleClickEvent (const ClickEvent &evt)

Handle a click event.

virtual void handleDragEvent (const DragEvent &evt)

Handle drag events.

· virtual void handleGestureEvent (const GestureEvent &evt)

Handle gestures.

virtual void handleTickEvent ()

Handle tick.

virtual void handleKeyEvent (uint8_t c)

Handle an incoming character received by the HAL layer.

virtual void handlePendingScreenTransition ()

Evaluates the pending Callback instances.

virtual void cacheDrawOperations (bool enableCache)

This functions allows for deferring draw operations to a later time.

void registerTimerWidget (Drawable *w)

Adds a widget to the list of widgets receiving ticks.

• void clearAllTimerWidgets ()

Clears all currently registered timer widgets.

void unregisterTimerWidget (const Drawable *w)

Removes a widget from the list of widgets receiving ticks.

• uint16_t getNumberOfRegisteredTimerWidgets () const

gets the number of timer widgets that has been registered

uint16_t getTimerWidgetCountForDrawable (const Drawable *w) const

Gets the number of timer events registered to a widget.

Static Public Member Functions

static Application * getInstance ()

Gets the single instance application.

static void setDebugPrinter (DebugPrinter *printer)

Sets the DebugPrinter object to be used by the application.

static DebugPrinter * getDebugPrinter ()

Returns the DebugPrinter object associated with the application.

static void setDebugString (const char *string)

Sets the debug string to be displayed onscreen.

Static Public Attributes

• static const uint8_t MAX_TIMER_WIDGETS = 32

Maximum number of widgets receiving ticks.

• static const uint16_t TICK_INTERVAL_MS = 10

Deprecated, do not use this constant. Tick interval depends on VSYNC of your target platform.

Protected Member Functions

· void invalidateArea (Rect area)

Invalidates this area.

• Application ()

Procected constructor.

Protected Attributes

Vector < Drawable *, MAX_TIMER_WIDGETS > timerWidgets

List of widgets that receive timer ticks.

uint8_t timerWidgetCounter [MAX_TIMER_WIDGETS]

A counter for each potentially registered timer widget. Increase when registering for timer events, decrease when unregistering.

Vector < Rect, 8 > cachedDirtyAreas

When draw caching is enabled, these rects keeps track of the dirty screen area.

• Vector< Rect, 8 > lastRects

The dirty areas from last frame that needs to be redrawn because we have swapped frame buffers.

bool drawCacheEnabled

True when draw caching is active.

· bool transitionHandled

True if the transition is done and Screen::afterTransition has been called.

Static Protected Attributes

static Screen * currentScreen

Pointer to currently displayed Screen.

static Transition * currentTransition

Pointer to current transition.

- static Application * instance
- static DebugPrinter * debugPrinter

Pointer to the DebugPrinter instance.

7.24.1 Detailed Description

The Application class is the main interface for manipulating screen contents. It holds a pointer to the currently displayed Screen, and delegates draw requests and events to that Screen. Additionally it contains some global application settings.

A user-defined application subclass can be defined to override standard functionality.

See also

UIEventListener

7.24.2 Constructor & Destructor Documentation

7.24.2.1 Application()

```
Application ( ) [protected]
```

Procected constructor.

7.24.3 Member Function Documentation

7.24.3.1 appSwitchScreen()

An application specific function for switching screen. Overloading this can provide a means to switch screen from places that does not have access to a pointer to the new screen. Base implementation is empty.

Parameters

screen←	An id that maps to the desired screen.
ld	

7.24.3.2 cacheDrawOperations()

```
void cacheDrawOperations (
          bool enableCache ) [virtual]
```

This functions allows for deferring draw operations to a later time. If active, calls to draw will simply note that the specified area is dirty, but not perform any actual drawing. When disabling the draw cache, the dirty area will be flushed (drawn) immediately.

Parameters

enableCache	if true, all future draw operations will be cached. If false draw caching is disabled, and the
	current cache (if not empty) is drawn immediately.

7.24.3.3 clearAllTimerWidgets()

```
void clearAllTimerWidgets ( )
```

Clears all currently registered timer widgets.

```
7.24.3.4 draw() [1/2] void draw ( ) [virtual]
```

Initiate a draw operation of the entire screen. Standard implementation is to delegate draw request to the current Screen.

Initiate a draw operation of the specified region of the screen. Standard implementation is to delegate draw request to the current Screen.

Note

Unlike Widget::draw this is safe to call from user code as it will properly traverse widgets in z-order. The coordinates given must be absolute coordinates.

Parameters

in rect The area to draw.

7.24.3.6 getCurrentScreen()

```
Screen * getCurrentScreen ( ) [inline]
```

Gets the current screen.

Returns

The current screen.

7.24.3.7 getDebugPrinter()

```
static const DebugPrinter * getDebugPrinter ( ) [inline], [static]
```

Returns the DebugPrinter object associated with the application.

Returns

DebugPrinter The DebugPrinter object.

7.24.3.8 getInstance()

```
static Application * getInstance ( ) [static]
```

Gets the single instance application.

Returns

The instance of this application.

7.24.3.9 getNumberOfRegisteredTimerWidgets()

```
\verb|uint16_t| getNumberOfRegisteredTimerWidgets ( ) const|\\
```

gets the number of timer widgets that has been registered.

Returns

The size of timerWidgets.

7.24.3.10 getTimerWidgetCountForDrawable()

Gets the number of timer events registered to a widget, i.e. how many times a drawable must be unregistered until it no longer receives timer ticks.

Parameters

w The widget to to get count from.

Returns

0 if the drawable is not registered as a timer widget, otherwise returns how many times the drawable is currently registered.

7.24.3.11 handleClickEvent()

Handle a click event. Standard implementation is to delegate the event to the current screen. Called by the framework when a click is detected by some platform specific means.

Parameters

```
evt The ClickEvent.
```

Reimplemented from UIEventListener.

7.24.3.12 handleDragEvent()

Handle drag events. Called by the framework when a drag is detected by some platform specific means. Standard implementation is to delegate drag event to current screen.

Parameters

```
evt The drag event, expressed in absolute coordinates.
```

Reimplemented from UIEventListener.

7.24.3.13 handleGestureEvent()

```
void handleGestureEvent ( {\tt const~GestureEvent~\&~evt~)} \quad [{\tt virtual}]
```

Handle gestures. Called by the framework when a gesture is detected by some platform specific means. Standard implementation is to delegate drag event to current screen.

Parameters

```
evt The gesture event.
```

Reimplemented from UIEventListener.

7.24.3.14 handleKeyEvent()

```
void handleKeyEvent ( \mbox{uint8\_t} \ c \ ) \ \ \mbox{[virtual]}
```

Handle an incoming character received by the HAL layer. Standard implementation delegates to current screen (which, in turn, does nothing).

Parameters

c The incomming character to handle.

Reimplemented from UIEventListener.

7.24.3.15 handlePendingScreenTransition()

```
void handlePendingScreenTransition ( ) [virtual]
```

Evaluates the pending Callback instances. If a callback is valid, it is executed and a Screen transition is executed. This base implementation is empty and does nothing.

Reimplemented from UIEventListener.

Reimplemented in MVPApplication.

7.24.3.16 handleTickEvent()

```
void handleTickEvent ( ) [virtual]
```

Handle tick. Standard implementation is to delegate tick to the widgets that have registered to receive one. Called by some platform specific means.

Reimplemented from UIEventListener.

7.24.3.17 invalidateArea()

Invalidates this area.

Parameters

area The area to invalidate.

7.24.3.18 registerTimerWidget()

Adds a widget to the list of widgets receiving ticks every frame (typically 16.67ms)

Note

The framework keeps track of the number of times a specific widget is registered.

Parameters

in	W	The widget to add.

See also

unregisterTimerWidget

7.24.3.19 setDebugPrinter()

Sets the DebugPrinter object to be used by the application to print debug messages.

Parameters

in printer The debug printer to configu

7.24.3.20 setDebugString()

Sets the debug string to be displayed onscreen on top of the framebuffer.

Parameters

in	string	The debug string to display onscreen.	1
----	--------	---------------------------------------	---

7.24.3.21 switchScreen()

Switch to another Screen. Will call tearDownScreen on current Screen before switching, and subsequently call setupScreen and draw automatically for the new Screen.

Parameters

in	newScreen	A pointer to the new screen.
----	-----------	------------------------------

7.24.3.22 unregisterTimerWidget()

Removes a widget from the list of widgets receiving ticks every frame (typically 16.67ms) milliseconds.

Note

If widget has been registered multiple times, an equal number of calls to unregister are required to stop widget from receiving tick events.

Parameters

	in	W	The widget to remove.
--	----	---	-----------------------

7.24.4 Member Data Documentation

7.24.4.1 instance

```
Application* instance [static], [protected]
```

Pointer to the instance of the Application-derived subclass.

Note

Must be set by subclass constructor!

7.24.4.2 MAX_TIMER_WIDGETS

```
const uint8_t MAX_TIMER_WIDGETS = 32 [static]
```

Remarks

Memory impact: x * (sizeof(Drawable*)+1)

7.25 Bitmap Class Reference

This class provides a proxy object for a bitmap image.

```
#include <touchgfx/Bitmap.hpp>
```

Classes

struct BitmapData

Data of a bitmap.

struct CacheTableEntry

Cache bookkeeping.

struct DynamicBitmapData

Data of a dynamic bitmap.

Public Types

 enum ClutFormat { CLUT_FORMAT_L8_ARGB8888, CLUT_FORMAT_L8_RGB888, CLUT_FORMAT_L8 — RGB565 }

Color data of a clut can be stored in the following formats.

enum BitmapFormat {

RGB565, RGB888, ARGB8888, BW, BW_RLE, GRAY2, GRAY4, ARGB2222, ABGR2222, RGBA2222, BGRA2222, L8 }

Data of a bitmap can be stored in the following formats.

Public Member Functions

Bitmap (const BitmapId id=BITMAP_INVALID)

Creates and binds a Bitmap instance to the corresponding entry in the BitmapData array.

· Bitmapld getId () const

Gets the id of this Bitmap.

const uint8_t * getData () const

Gets a pointer to the Bitmap data.

const uint8_t * getAlphaData () const

Gets a pointer to the alpha data, if present in the bitmap.

• const uint8_t * getExtraData () const

Gets a pointer to the extra (alpha) data, if present in the bitmap.

· BitmapFormat getFormat () const

Gets the format of how the bitmap is stored.

• uint16_t getWidth () const

Gets the width of the Bitmap in pixels.

uint16_t getHeight () const

Gets the height of the Bitmap in pixels.

Rect getRect () const

Gets the rectangle describing the dimensions of the Bitmap.

· bool isAlphaPerPixel () const

Query if this object has an alpha channel.

• Rect getSolidRect () const

Gets the largest solid rectangle in the bitmap.

bool hasTransparentPixels () const

Query if this object has transparent pixels.

bool operator== (const Bitmap &other) const

Equality operator.

• bool operator!= (const Bitmap &other) const

Inequality operator.

Static Public Member Functions

static void registerBitmapDatabase (const BitmapData *data, const uint16_t n, uint16_t *cachep=0, uint32_t csize=0, uint32_t numberOfDynamicBitmaps=0)

Registers an array of bitmaps.

• static bool cache (Bitmapld id)

Cache this bitmap into RAM.

• static bool cacheReplaceBitmap (BitmapId out, BitmapId in)

Replace a bitmap in RAM with another Bitmap.

• static bool cacheRemoveBitmap (Bitmapld id)

Remove this bitmap from the RAM cache.

static uint8_t * cacheGetAddress (BitmapId id)

Get address of cache buffer for this bitmap.

• static bool cachelsCached (Bitmapld id)

Check if the Bitmap is cached.

static bool cacheAll ()

Cache all bitmaps from the Bitmap Database into RAM.

• static void clearCache ()

Clears the cached bitmaps from RAM.

 static BitmapId dynamicBitmapCreate (const uint16_t width, const uint16_t height, BitmapFormat format, ClutFormat clutFormat=CLUT_FORMAT_L8_ARGB8888)

Create a dynamic bitmap.

static bool dynamicBitmapDelete (BitmapId id)

Delete a dynamic bitmap.

static uint8_t * dynamicBitmapGetAddress (BitmapId id)

Get the address of the dynamic bitmap data.

static bool dynamicBitmapSetSolidRect (BitmapId id, const Rect &solidRect)

Set the solid rectangle of a dynamic bitmap.

• static bool dynamicBitmapAddSolidRect (BitmapId id, const Rect &solidRect)

Updates the solid rectangle of a dynamic bitmap.

static void setCache (uint16_t *cachep, uint32_t csize, uint32_t numberOfDynamicBitmaps=0)

Register a memory region in which bitmap data can be cached.

• static void removeCache ()

Removes the bitmap cache.

static uint8_t * getCacheTopAddress ()

Gets the address of the first unused memory in the cache.

static void compactCache ()

Compact the bitmap cache to get continuous free memory on top.

7.25.1 Detailed Description

This class provides a proxy object for a bitmap image stored in the application specific bitmap database. The proxy provides access to the raw bitmap data as well as metadata.

7.25.2 Member Enumeration Documentation

7.25.2.1 BitmapFormat

enum BitmapFormat

Data of a bitmap can be stored in the following formats.

Enumerator

RGB565	16-bit, 5 bits for red, 6 bits for green, 5 bits for blue, no alpha channel
RGB888	24-bit, 8 bits for each of red, green and blue, no alpha channel
ARGB8888	32-bit, 8 bits for each of red, green, blue and alpha channel
BW	1-bit, black / white, no alpha channel
BW_RLE	1-bit, black / white, no alpha channel compressed with horizontal RLE
GRAY2	2-bit grayscale
GRAY4	4-bit grayscale
ARGB2222	8-bit color
ABGR2222	8-bit color
RGBA2222	8-bit color
BGRA2222	8-bit color
L8	8-bit indexed color

7.25.2.2 ClutFormat

```
enum ClutFormat
```

Color data of a clut can be stored in the following formats.

Enumerator

CLUT_FORMAT_L8_ARGB8888	32-bit, 8 bits for each of red, green, blue and per pixel alpha channel
CLUT_FORMAT_L8_RGB888	24-bit, 8 bits for each of red, green and blue, no per pixel alpha channel
CLUT_FORMAT_L8_RGB565	16-bit, 5 bits for red, 6 bits for green, 5 bits for blue, no per pixel alpha channel

7.25.3 Constructor & Destructor Documentation

Creates and binds a Bitmap instance to the corresponding entry in the BitmapData array.

Parameters

id The unique bitmap identifier.

7.25.4 Member Function Documentation

Cache this bitmap into unused RAM in the bitmap cache.

Note

A memory region large enough to hold this bitmap must be configured and a large enough part of it must be available. Caching of a bitmap may involve a defragmentation of the bitmap cache. See TouchGFX documentation for details on caching.

Parameters

id The id of the bitmap to cache.

Returns

true if caching went well, false otherwise.

See also

registerBitmapDatabase

7.25.4.2 cacheAll()

```
static bool cacheAll ( ) [static]
```

Cache all bitmaps from the Bitmap Database into RAM.

Note

A memory region large enough to hold all bitmaps must be configured. See TouchGFX documentation for details on caching.

Returns

True if all bitmaps where cached.

See also

cache.

7.25.4.3 cacheGetAddress()

```
static uint8_t * cacheGetAddress ( {\tt BitmapId} \ id \ ) \quad [{\tt static}]
```

Get address of cache buffer for this bitmap. Note: The address is only valid until next Bitmap::cache() call.

Parameters

id The id of the bitmap in cache.

Returns

Address if bitmap was found, zero otherwise.

7.25.4.4 cachelsCached()

```
static bool cacheIsCached ( {\tt BitmapId} \ id \ ) \quad [{\tt static}]
```

Check if the Bitmap is cached.

Parameters

id The id of the bitmap.

Returns

true if bitmap is cached.

7.25.4.5 cacheRemoveBitmap()

```
static bool cacheRemoveBitmap ( {\tt BitmapId} \ id \ ) \quad [{\tt static}]
```

Remove this bitmap from the RAM cache.

Note

The bitmap will be removed from the RAM cache. Unless the bitmap is otherwise stored in (slow) RAM it can not be drawn anymore and must be cached again before use. The RAM freed can be used for caching of another bitmap. See TouchGFX documentation for details on caching.

Parameters

id The id of the bitmap to cache.

Returns

true if bitmap was found and removed, false otherwise.

See also

registerBitmapDatabase

7.25.4.6 cacheReplaceBitmap()

Replace a bitmap in RAM with another Bitmap. The Bitmaps must have same size.

Parameters

out	The id of the bitmap to remove from the cache.
in	The id of the bitmap to cache.

Returns

true if the replacement went well, false otherwise.

7.25.4.7 clearCache()

```
static void clearCache ( ) [static]
```

Clears the cached bitmaps from RAM.

7.25.4.8 compactCache()

```
static void compactCache ( ) [static]
```

Compact the bitmap cache to get continuous free memory on top.

Note

This method is called by Bitmap::cache when required.

7.25.4.9 dynamicBitmapAddSolidRect()

Updates the solid rectangle of a dynamic bitmap to include the given rectangle. Only relevant for ARGB8888 bitmaps and 8bpp bitmap formats, as these formats include an alpha channel. The solid part of the bitmap is drawn faster than the transparent parts.

Parameters

id	The identifier.
solidRect	The solid rectangle.

Returns

true if it succeeds, false if it fails.

7.25.4.10 dynamicBitmapCreate()

Create a dynamic bitmap. The clutFormat parameter is ignored for bitmaps not in L8 format.

Parameters

width	Width of the bitmap.
height	Height of the bitmap.
format	Bitmap format of the bitmap.
clutFormat	Color lookup table format of the bitmap.

Returns

BitmapId of the new bitmap or BITMAP_INVALID if cache memory is full.

Note

Creation of a new dynamic bitmap may cause existing dynamic bitmaps to be moved in memory. Do not rely on bitmap memory addresses of dynamic bitmaps obtained from dynamicBitmapGetAddress() to be valid across calls to dynamicBitmapCreate().

See also

dynamicBitmapAddress

7.25.4.11 dynamicBitmapDelete()

Delete a dynamic bitmap.

Parameters

id The BitmapId of the dynamic bitmap.

Returns

true if it succeeds, false if it fails.

7.25.4.12 dynamicBitmapGetAddress()

Get the address of the dynamic bitmap data. It is important that the address of a dynamic bitmap is not stored elsewhere as a dynamic bitmap may be moved in memory when other bitmaps are added and removed. Only store the BitmapId and ask for the address of the bitmap data when needed.

Parameters

id The Bitmapld of the dynamic bitmap.

Returns

null if it fails, else an uint8_t*.

Note

The address of a dynamic bitmap may change when other dynamic bitmaps are added and removed. Never store the address of dynamic images, only store the Bitmapld as that will not change.

7.25.4.13 dynamicBitmapSetSolidRect()

```
static bool dynamicBitmapSetSolidRect (
```

```
BitmapId id,
const Rect & solidRect ) [static]
```

Set the solid rectangle of a dynamic bitmap. Only relevant for ARGB8888 bitmaps and 8bpp bitmap formats, as these formats include an alpha channel. The solid part of the bitmap is drawn faster than the transparent parts.

Parameters

id	The identifier.
solidRect	The solid rectangle.

Returns

true if it succeeds, false if it fails.

7.25.4.14 getAlphaData()

```
const uint8_t * getAlphaData ( ) const
```

Gets a pointer to the alpha data, if present in the bitmap. For images stored in L8 format, a pointer to the CLUT will be returned. For non-opaque RGB565 images, a pointer to the alpha channel will be returned.

Note

If this bitmap is cached, it will return the cached version of alpha data for this bitmap.

Returns

A pointer to the raw alpha channel data or CLUT. If no alpha channel or CLUT exist for the given Bitmap, 0 is returned.

See also

getExtraData

7.25.4.15 getCacheTopAddress()

```
static void getCacheTopAddress ( ) [inline], [static]
```

Gets the address of the first unused memory in the cache. Can be used in advanced application to reduce power consumption of external RAM by turning off unused RAM.

Returns

Returns the highest used address in the cache.

7.25.4.16 getData()

```
const uint8_t * getData ( ) const
```

Gets a pointer to the Bitmap data.

Note

If this bitmap is cached, it will return the cached version of bitmap data.

Returns

A pointer to the raw bitmap data.

7.25.4.17 getExtraData()

```
const uint8_t * getExtraData ( ) const
```

Gets a pointer to the extra (alpha) data, if present in the bitmap. For images stored in L8 format, a pointer to the CLUT will be returned. For non-opaque RGB565 images, a pointer to the alpha channel will be returned.

Note

If this bitmap is cached, it will return the cached version of alpha data for this bitmap.

Returns

A pointer to the raw alpha channel data or CLUT. If no alpha channel or CLUT exist for the given Bitmap, 0 is returned.

7.25.4.18 getFormat()

```
BitmapFormat getFormat ( ) const
```

Gets the format of how the bitmap is stored.

Returns

The format of how the bitmap data is stored.

7.25.4.19 getHeight()

```
uint16_t getHeight ( ) const
```

Gets the height of the Bitmap in pixels.

Returns

The bitmap height in pixels.

7.25.4.20 getId()

```
BitmapId getId ( ) const [inline]
```

Gets the id of this Bitmap.

Returns

The id of this Bitmap.

7.25.4.21 getRect()

```
Rect getRect ( ) const [inline]
```

Gets the rectangle describing the dimensions of the Bitmap.

Returns

a Rect describing the dimensions of this bitmap.

7.25.4.22 getSolidRect()

```
Rect getSolidRect ( ) const
```

Gets the largest solid, i.e. not transparent, rectangle in the bitmap.

Returns

The maximum solid rectangle of the bitmap.

7.25.4.23 getWidth()

```
uint16_t getWidth ( ) const
```

Gets the width of the Bitmap in pixels.

Returns

The bitmap width in pixels.

7.25.4.24 hasTransparentPixels()

```
bool hasTransparentPixels ( ) const
```

Returns

True if this bitmap has transparent pixels.

7.25.4.25 isAlphaPerPixel()

```
bool isAlphaPerPixel ( ) const [inline]
```

Query if this object has an alpha channel.

Returns

True if the bitmap contains an alpha channel (an alpha value for each pixel)

7.25.4.26 operator"!=()

Inequality operator.

Parameters

other The bitmap to comp	are with.
--------------------------	-----------

Returns

True if this bitmap has a different id than the other bitmap.

7.25.4.27 operator==()

Equality operator.

Parameters

other

Returns

True if this bitmap has the same id as the other bitmap.

7.25.4.28 registerBitmapDatabase()

Registers an array of bitmaps. All Bitmap instances are bound to this database.

Parameters

	data	A reference to the BitmapData storage array.
	n	The number of bitmaps in the array.
in,out	cachep	(Optional) Pointer to memory region in which bitmap data can be cached.
	csize	Size of cache memory region in bytes (0 if unused)
	numberOfDynamicBitmaps	Number of dynamic bitmaps to be allowed in the cache.

7.25.4.29 removeCache()

```
static void removeCache ( ) [static]
```

Removes the bitmap cache. The memory can hereafter be used for other purposes. All dynamic bitmap IDs are not valid after this.

7.25.4.30 setCache()

Register a memory region in which bitmap data can be cached.

Parameters

in,out	cachep	Pointer to memory region in which bitmap data can be cached.
	csize	Size of cache memory region in bytes.
	numberOfDynamicBitmaps	Number of dynamic bitmaps to be allowed in the cache.

7.26 Bitmap::BitmapData Struct Reference

Data of a bitmap.

```
#include <touchgfx/Bitmap.hpp>
```

Public Member Functions

· BitmapFormat getFormat () const

Gets the format.

Public Attributes

· const uint8 t *const data

The data of this bitmap.

const uint8_t *const extraData

The data of either the alpha channel if exist or clut data in case of indexed color bitmap (contains 0 if no alpha channel neither clut exist)

· const uint16_t width

The width of the bitmap.

· const uint16_t height

The height of the bitmap.

const uint16_t solidRect_x
 The x coordinate of the maximum solid rectangle of the bitmap.

· const uint16 t solidRect y

The y coordinate of the maximum solid rectangle of the bitmap.

const uint16_t solidRect_width: 13

The width of the maximum solid rectangle of the bitmap.

· const uint16 t format hi: 3

Determine the format of the data (high 3 bits)

· const uint16_t solidRect_height: 13

The height of the maximum solid rectangle of the bitmap.

const uint16_t format_lo: 3

Determine the format of the data (low 3 bits)

7.26.1 Detailed Description

Data of a bitmap.

7.26.2 Member Function Documentation

```
7.26.2.1 getFormat()
```

```
BitmapFormat getFormat ( ) const [inline]
```

Gets the format by combining the high and low parts (format |h| << 3) | format |h|

Returns

The format.

7.27 BlitOp Struct Reference

BlitOp instances carry the required information for performing operations on the LCD (frame buffer) using DMA.

```
#include <touchgfx/hal/BlitOp.hpp>
```

Public Attributes

• uint32_t operation

The operation to perform.

const uint16_t * pSrc

Pointer to the source (pixels or indexes)

const uint8_t * pClut

Pointer to the source CLUT entires.

uint16_t * pDst

Pointer to the destination.

· uint16_t nSteps

The number of pixels in a line.

• uint16_t nLoops

The number of lines.

uint16_t srcLoopStride

The number of bytes to stride the source after every loop.

uint16_t dstLoopStride

The number of bytes to stride the destination after every loop.

· colortype color

Color to fill.

• uint8_t alpha

The alpha to use.

7.28 Box Class Reference 171

uint8_t srcFormat

The source format.

uint8_t dstFormat

The destination format.

7.27.1 Detailed Description

BlitOp instances carry the required information for performing operations on the LCD (frame buffer) using DMA.

7.27.2 Member Data Documentation

7.27.2.1 dstFormat

uint8_t dstFormat

See also

BitmapFormat

7.27.2.2 operation

uint32_t operation

See also

BlitOperations

7.27.2.3 srcFormat

uint8_t srcFormat

See also

BitmapFormat

7.28 Box Class Reference

Simple widget capable of showing a rectangle of a specific color and an optional alpha.

#include <touchgfx/widgets/Box.hpp>

Public Member Functions

• Box ()

Constructor.

Box (uint16_t width, uint16_t height, colortype color, uint8_t alpha=255)

Constructor.

virtual ∼Box ()

Destructor.

virtual Rect getSolidRect () const

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent).

void setColor (colortype color)

Sets the color of the rectangle.

• colortype getColor () const

Gets the current color of the Box.

void setAlpha (uint8_t alpha)

Sets the alpha value for this Box.

• uint8_t getAlpha () const

Returns the current alpha value.

· virtual void draw (const Rect &area) const

Draws the box.

void forceReportAsSolid (bool solid)

Override solid area for the Box.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

• uint8_t alpha

The alpha value used for this Box.

· colortype color

The fill color for this Box.

· bool reportAsSolid

Additional Inherited Members

7.28.1 Detailed Description

Simple widget capable of showing a rectangle of a specific color and an optional alpha.

See also

Widget

7.28.2 Constructor & Destructor Documentation

```
7.28.2.1 Box() [1/2]
```

```
Box ( ) [inline]
```

Constructs a new Box with a default alpha value of 255 (solid)

7.28 Box Class Reference 173

Construct a Box.

Parameters

width	The width of the box.
height	The height of the box.
color	The color of the box.
alpha	The alpha of the box. Default is 255 (solid).

```
7.28.2.3 ~Box()
~Box ( ) [inline], [virtual]
Destructor.
```

7.28.3 Member Function Documentation

Draws the Box.

Parameters

area	The rectangle to draw, with coordinates relative to the containing layer.

Implements Drawable.

7.28.3.2 forceReportAsSolid()

```
void forceReportAsSolid (
                bool solid ) [inline]
```

If this is set, getSolidRect() will report the widget as completely solid even if is (semi-)transparent.

Note

Very rarely used in practice.

Parameters

solid

true if this Box should report as solid, even when not.

7.28.3.3 getAlpha()

```
uint8_t getAlpha ( ) const [inline]
```

Returns

Gets the current alpha value of the Box.

7.28.3.4 getColor()

```
colortype getColor ( ) const [inline]
```

Gets the current color of the Box.

Returns

The current color.

7.28.3.5 getSolidRect()

```
virtual Rect getSolidRect ( ) const [virtual]
```

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent). Used by JSMOC to prune the draw graph.

Note

The rectangle returned must be relative to (0, 0), meaning that to indicate a completely solid widget, Rect(0, 0, getWidth(), getHeight()) must be returned.

Returns

The solid rect.

Implements Drawable.

7.28.3.6 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_BOX.

Reimplemented from Widget.

7.28.3.7 setAlpha()

Sets the alpha value for this Box.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.28.3.8 setColor()

Sets the color of the rectangle.

Parameters

color The color of the	e box.
------------------------	--------

7.28.4 Member Data Documentation

7.28.4.1 reportAsSolid

```
bool reportAsSolid [protected]
```

See also

forceReportAsSolid.

7.29 BoxProgress Class Reference

A box progress.

#include <touchgfx/containers/progress_indicators/BoxProgress.hpp>

Public Member Functions

• BoxProgress ()

Default constructor.

virtual ∼BoxProgress ()

Destructor.

- virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)
 - Sets the position and dimension of the box progress indicator.
- virtual void setColor (colortype color)

Sets the color.

· virtual colortype getColor () const

Gets the color.

• virtual void setAlpha (uint8_t alpha)

Sets the alpha.

• virtual uint8_t getAlpha () const

Gets the alpha.

virtual void setValue (int value)

Sets a value.

Protected Attributes

Box box

The box.

Additional Inherited Members

7.29.1 Detailed Description

A Box progress which shows the current progress using a simple Box. It is possible to set the color and the alpha of the box. It is also possible to control in what direction the box will progress (up, down, to the left or to the right).

See also

Box

7.29.2 Constructor & Destructor Documentation

```
7.29.2.1 BoxProgress()
```

```
BoxProgress ( )
```

Default constructor.

7.29.2.2 \sim BoxProgress()

```
\simBoxProgress ( ) [virtual]
```

Destructor.

7.29.3 Member Function Documentation

```
7.29.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const [virtual]
```

Gets the alpha of the Box.

Returns

The alpha.

```
See also
```

Box

```
7.29.3.2 getColor()
```

```
colortype getColor ( ) const [virtual]
```

Gets the color of the Box.

Returns

The color.

See also

Box

7.29.3.3 setAlpha()

Sets the alpha of the Box.

Parameters

```
alpha The alpha.
```

See also

Box

7.29.3.4 setColor()

Sets the color of the Box.

Parameters

```
color The color.
```

See also

Box

7.29.3.5 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the box progress indicator relative to the background image.

Parameters

X	The x coordinate.
У	The y coordinate.
width The width of the box progress indicator.	
height	The height of the box progress indicator.

Reimplemented from AbstractProgressIndicator.

7.29.3.6 setValue()

```
virtual void setValue ( int \ value \ ) \quad [virtual]
```

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

value	The value.
-------	------------

Reimplemented from AbstractProgressIndicator.

7.30 BoxWithBorder Class Reference

A box with border.

```
#include <touchgfx/containers/buttons/BoxWithBorder.hpp>
```

Public Member Functions

• BoxWithBorder ()

Default constructor.

 BoxWithBorder (uint16_t width, uint16_t height, colortype color, colortype borderColor, uint8_t borderSize, uint8_t alpha=255)

Constructor.

virtual ∼BoxWithBorder ()

Destructor.

virtual Rect getSolidRect () const

Gets solid rectangle.

• void setColor (colortype color)

Sets a color.

• colortype getColor () const

Gets the color.

void setBorderColor (colortype color)

Sets border color.

• colortype getBorderColor () const

Gets border color.

• void setBorderSize (uint8_t size)

Sets border size.

• uint8_t getBorderSize () const

Gets border size.

void setAlpha (uint8_t alpha)

Sets an alpha.

• uint8_t getAlpha () const

Gets the alpha.

· virtual void draw (const Rect &area) const

Draws the given area.

virtual uint16_t getType () const

Gets the type.

Protected Attributes

• uint8_t alpha

The alpha.

· colortype color

The color.

• colortype borderColor

The border color.

· uint8 t borderSize

Size of the border.

Additional Inherited Members

7.30.1 Constructor & Destructor Documentation

7.30.1.1 BoxWithBorder()

```
BoxWithBorder (
          uint16_t width,
          uint16_t height,
          colortype color,
          colortype borderColor,
          uint8_t borderSize,
          uint8_t alpha = 255 ) [inline]
```

Parameters

width	The width.
height	The height.
color	The color.
borderColor	The border color.
borderSize	Size of the border.

TO BURGEX V4.12.3 (C) OPHING PANIC TO BE CALDING 2014-2019

7.30.2 Member Function Documentation

```
7.30.2.1 draw()
void draw (
             const Rect & area ) const [virtual]
Parameters
 area The area.
Implements Drawable.
7.30.2.2 getAlpha()
uint8_t getAlpha ( ) const [inline]
Returns
     The alpha.
7.30.2.3 getBorderColor()
colortype getBorderColor ( ) const [inline]
Returns
     The border color.
7.30.2.4 getBorderSize()
uint8_t getBorderSize ( ) const [inline]
Returns
     The border size.
7.30.2.5 getColor()
colortype getColor ( ) const [inline]
```

Returns

The color.

```
7.30.2.6 getSolidRect()
```

```
Rect getSolidRect ( ) const [virtual]
```

Returns

The solid rectangle.

Implements Drawable.

```
7.30.2.7 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

Returns

The type.

Reimplemented from Widget.

7.30.2.8 setAlpha()

Parameters

```
alpha The alpha.
```

7.30.2.9 setBorderColor()

Parameters

```
color The color.
```

7.30.2.10 setBorderSize()

Parameters

```
size The size.
```

7.30.2.11 setColor()

Parameters

color The color.

7.31 BoxWithBorderButtonStyle < T > Class Template Reference

A box with border button style.

#include <touchgfx/containers/buttons/BoxWithBorderButtonStyle.hpp>

Public Member Functions

• BoxWithBorderButtonStyle ()

Default constructor.

virtual ~BoxWithBorderButtonStyle ()

Destructor.

• void setBoxWithBorderPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the size and position of this BoxWithBorderButtonStyle.

void setBoxWithBorderWidth (int16 t width)

Sets a width.

void setBoxWithBorderHeight (int16_t height)

Sets a height.

void setBoxWithBorderColors (const colortype colorReleased, const colortype colorPressed, const colortype borderColorReleased, const colortype borderColorPressed)

Sets the colors.

• void setBorderSize (uint8_t size)

Sets border size.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· BoxWithBorder borderBox

The border box.

· colortype up

The up.

· colortype down

The down.

colortype borderUp

The border up.

· colortype borderDown

The border down.

7.31.1 Detailed Description

 $\label{template} \begin{tabular}{ll} template < class T > \\ class touchgfx::BoxWithBorderButtonStyle < T > \\ \end{tabular}$

A box with border button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show a box with a border in different colors depending on the state of the button (pressed or released).

An image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show one of two images depending on the state of the button (pressed or released).

Template Parameters

T Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer, BoxWithBorder

7.31.2 Member Function Documentation

7.31.2.1 setBorderSize()

Parameters

```
size The size.
```

7.31.2.2 setBoxWithBorderColors()

Parameters

colorReleased	The color released.
colorPressed	The color pressed.
borderColorReleased	The border color released.
borderColorPressed	The border color pressed.

7.31.2.3 setBoxWithBorderHeight()

Parameters

height The height.

7.31.2.4 setBoxWithBorderPosition()

```
void setBoxWithBorderPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [inline]
```

Sets the size and position of this BoxWithBorderButtonStyle, relative to its parent.

Note

Changing this does not automatically yield a redraw.

Parameters

X	The x coordinate of this BoxWithBorderButtonStyle.	
У	The y coordinate of this BoxWithBorderButtonStyle.	
width	The width of this BoxWithBorderButtonStyle.	
height	The height of this BoxWithBorderButtonStyle.	

7.31.2.5 setBoxWithBorderWidth()

Parameters

width	The width.
-------	------------

7.32 Button Class Reference

A button with two states.

```
#include <touchgfx/widgets/Button.hpp>
```

Public Member Functions

• Button ()

Default constructor.

virtual ∼Button ()

Destructor.

virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

• virtual void setBitmaps (const Bitmap &bmpReleased, const Bitmap &bmpPressed)

Sets the bitmaps used by this button.

· virtual Rect getSolidRect () const

Gets solid rectangle.

· void setAlpha (uint8_t alpha)

Sets the alpha value for the image.

• uint8_t getAlpha () const

Gets the current alpha value.

• Bitmap getCurrentlyDisplayedBitmap () const

Gets currently displayed bitmap.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

· Bitmap up

The image to display when button is released.

· Bitmap down

The image to display when button is pressed.

• uint8_t alpha

The current alpha value. 255 denotes solid, 0 denotes completely transparent.

Additional Inherited Members

7.32.1 Detailed Description

A button consists of two images, one for its normal state and one when it is pressed down.

See also

AbstractButton

7.32.2 Constructor & Destructor Documentation

```
7.32.2.1 Button()
Button ( ) [inline]
Default constructor.

7.32.2.2 ~Button()
~Button ( ) [inline], [virtual]
Destructor.
```

7.32.3 Member Function Documentation

7.32.3.1 draw()

Parameters

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.
-----------------	--

See also

Drawable::draw()

Implements Drawable.

Reimplemented in ButtonWithLabel, and ButtonWithIcon.

7.32.3.2 getAlpha()

```
uint8_t getAlpha ( ) const [inline]
```

Gets the current alpha value.

Returns

The current alpha value.

7.32.3.3 getCurrentlyDisplayedBitmap()

```
Bitmap getCurrentlyDisplayedBitmap ( ) const [inline]
```

Function to obtain the currently displayed bitmap, which depends on the button's pressed state.

Returns

The bitmap currently displayed.

7.32.3.4 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets solid rectangle.

Returns

largest possible solid rect. Delegated to the largest solid rect of the button bitmap(s).

Implements Drawable.

Reimplemented in ButtonWithLabel.

7.32.3.5 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

```
TYPE BUTTON.
```

Reimplemented from AbstractButton.

Reimplemented in ButtonWithLabel, ButtonWithIcon, and ToggleButton.

7.32.3.6 setAlpha()

Sets the alpha value for the image.

Parameters

alpha	The alpha value. 255 = completely solid.
-------	--

7.32.3.7 setBitmaps()

Sets the bitmaps used by this button.

Parameters

bmpReleased	Bitmap to use when button is released.
bmpPressed	Bitmap to use when button is pressed.

Reimplemented in ToggleButton.

7.33 ButtonController Class Reference

Interface for sampling external key events.

#include <platform/driver/button/ButtonController.hpp>

Public Member Functions

- virtual ∼ButtonController ()
 - Destructor.
- virtual void init ()=0

Initializes button controller.

• virtual bool sample (uint8_t &key)=0

Sample external key events.

· virtual void reset ()

Resets button controller.

7.33.1 Detailed Description

Interface for sampling external key events.

7.33.2 Constructor & Destructor Documentation

```
7.33.2.1 ∼ButtonController()
~ButtonController ( ) [inline], [virtual]
```

7.33.3 Member Function Documentation

```
7.33.3.1 init()
void init ( ) [pure virtual]
Initializes button controller.
7.33.3.2 reset()
void reset ( ) [inline], [virtual]
Resets button controller. Does nothing in the default implementation.
```

```
7.33.3.3 sample()
bool sample (
             uint8_t & key ) [pure virtual]
```

Sample external key events.

Parameters

Destructor.

in,out	key	Output parameter that will be set to the key value if a keypress was detected.
--------	-----	--

Returns

True if a keypress was detected and the "key" parameter is set to a value.

7.34 Buttons Class Reference

A buttons.

```
#include <touchgfx/hal/Buttons.hpp>
```

Static Public Member Functions

· static void init ()

Perform configuration of IO pins.

static unsigned int sample ()

Sample button states.

7.34.1 Member Function Documentation

```
7.34.1.1 init()
static void init ( ) [static]
Perform configuration of IO pins.
7.34.1.2 sample()
static unsigned int sample ( ) [static]
Sample button states.
```

Returns

the sampled state of the buttons.

7.35 ButtonWithIcon Class Reference

A Button specialization that also displays an icon on top of the button bitmap.

```
#include <touchgfx/widgets/ButtonWithIcon.hpp>
```

Public Member Functions

 virtual void setBitmaps (const Bitmap &newBackgroundReleased, const Bitmap &newBackgroundPressed, const Bitmap &newIconReleased, const Bitmap &newIconPressed)

Sets the bitmaps used by this button.

void setIconX (int16_t x)

Sets the x coordinate of the icon bitmap.

void setlconY (int16 t y)

Sets the y coordinate of the icon bitmap.

void setIconXY (int16_t x, int16_t y)

Sets the x and y coordinates of the icon bitmap.

• Bitmap getCurrentlyDisplayedIcon () const

Function to obtain the currently displayed icon.

• int16_t getIconX () const

Gets the x coordinate of the icon bitmap.

• int16_t getIconY () const

Gets the y coordinate of the icon bitmap.

· virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

virtual uint16_t getType () const

For GUI testing only. Returns type of this drawable.

Protected Attributes

· Bitmap iconReleased

Icon to display when button is not pressed.

· Bitmap iconPressed

Icon to display when button is pressed.

int16_t iconX

x coordinate offset for icon.

int16 t iconY

y coordinate offset for icon.

Additional Inherited Members

7.35.1 Detailed Description

A Button specialization that also displays an icon on top of the button bitmap.

See also

Button

7.35.2 Member Function Documentation

```
7.35.2.1 draw()
```

Parameters

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.

See also

Drawable::draw()

Reimplemented from Button.

7.35.2.2 getCurrentlyDisplayedlcon()

```
Bitmap getCurrentlyDisplayedIcon ( ) const [inline]
```

Function to obtain the currently displayed icon, which depends on the button's pressed state.

Returns

The icon currently displayed.

7.35.2.3 getlconX()

```
int16_t getIconX ( ) const [inline]
```

Gets the x coordinate of the icon bitmap.

Returns

The x coordinate of the icon bitmap.

7.35.2.4 getlconY()

```
int16_t getIconY ( ) const [inline]
```

Gets the y coordinate of the icon bitmap.

Returns

The y coordinate of the icon bitmap.

7.35.2.5 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE BUTTONWITHICON.

Reimplemented from Button.

7.35.2.6 setBitmaps()

Sets the bitmaps used by this button.

Parameters

newBackgroundReleased	Bitmap to use when button is released.	
newBackgroundPressed	Bitmap to use when button is pressed.	
newlconReleased	The bitmap for the icon in the released/unpressed button state.	
newlconPressed	The bitmap for the icon in the pressed button state.	

7.35.2.7 setlconX()

```
void setIconX ( int16\_t \ x \ ) \quad [inline]
```

Sets the x coordinate of the icon bitmap.

Note

Changing this does not automatically yield a redraw.

The value will be overwritten by calling.

Parameters

x The new x value, relative to the background bitmap. A negative value is allowed.

7.35.2.8 setIconXY()

Sets the x and y coordinates of the icon bitmap.

Note

Changing this does not automatically yield a redraw.

Parameters

X	The new x value, relative to the background bitmap. A negative value is allowed.
У	The new y value, relative to the background bitmap. A negative value is allowed.

7.35.2.9 setlconY()

```
void setIconY ( int16\_t \ y \ ) \quad [inline]
```

Sets the y coordinate of the icon bitmap.

Note

Changing this does not automatically yield a redraw.

Parameters

y The new y value, relative to the background bitmap. A negative value is allowed.

7.36 ButtonWithLabel Class Reference

A Button specialization that also displays a text on top of the button bitmap.

#include <touchgfx/widgets/ButtonWithLabel.hpp>

Public Member Functions

ButtonWithLabel ()

Default constructor.

void setLabelText (TypedText t)

Sets the text to display on the button.

TypedText getLabelText () const

Gets the text used for the label.

void setLabelColor (colortype col, bool performInvalidate=false)

Sets label color.

void setLabelColorPressed (colortype col, bool performInvalidate=false)

Sets label color when the button is pressed.

void setLabelRotation (TextRotation rotation)

Sets the rotation of the text on the label.

• TextRotation getLabelRotation ()

Gets the current rotation of the text on the label.

void updateTextPosition ()

Positions the label text as horizontally centered.

• virtual Rect getSolidRect () const

Gets solid rectangle.

· virtual void draw (const Rect &area) const

Draws the given invalidated area.

• virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

TypedText typedText

The TypedText used for the button label.

colortype color

The color used for the label when not pressed.

colortype colorPressed

The color used for the label when pressed.

· TextRotation rotation

The rotation used for the label.

uint8_t textHeightIncludingSpacing

Total height of the label (text height + spacing).

Additional Inherited Members

7.36.1 Detailed Description

A Button specialization that also displays a text on top of the button bitmap.

See also

Button

7.36.2 Constructor & Destructor Documentation

```
7.36.2.1 ButtonWithLabel()
```

```
ButtonWithLabel ( )
```

Default constructor.

7.36.3 Member Function Documentation

```
7.36.3.1 draw()
```

Parameters

invalidatedArea The rectangle to draw, with coordinates relative to this drawable.

See also

Drawable::draw()

Reimplemented from Button.

7.36.3.2 getLabelRotation()

```
TextRotation getLabelRotation ( ) [inline]
```

Gets the current rotation of the text on the label.

Returns

The current rotation of the text.

7.36.3.3 getLabelText()

```
TypedText getLabelText ( ) const [inline]
```

Gets the text used for the label.

Returns

The text used for the label.

7.36.3.4 getSolidRect()

```
virtual Rect getSolidRect ( ) const [inline], [virtual]
```

Gets solid rectangle.

Returns

largest possible solid rect. Delegated to the largest solid rect of the button bitmap(s).

Reimplemented from Button.

7.36.3.5 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_BUTTONWITHLABEL.

Reimplemented from Button.

7.36.3.6 setLabelColor()

Sets label color.

Parameters

col	The color with which the text label should be drawn.	
performInvalidate Optional parameter. If true, performs an instant invalidation of the butt		

7.36.3.7 setLabelColorPressed()

Sets label color when the button is pressed.

Parameters

col The color with which the text label should be drawn when the button	
performInvalidate Optional parameter. If true, performs an instant invalidation of the button	

7.36.3.8 setLabelRotation()

Sets the rotation of the text on the label. Please note that this will not rotate the bitmap of the label, only the text.

Parameters

rotation	the rotation of the text. Default is TEXT_ROTATE_0.
----------	---

7.36.3.9 setLabelText()

Sets the text to display on the button. Texts with wildcards are not supported.

Parameters

```
t The text to display.
```

7.36.3.10 updateTextPosition()

```
void updateTextPosition ( ) [inline]
```

If the text changes due to a language change you may need to reposition the label text to stay horizontally centered.

Note

The method does not invalidate the button. This must be done manually.

7.37 CacheableContainer Class Reference

A CacheableContainer is a Container that can have its drawing redirected into a Bitmap.

```
#include <touchgfx/containers/CacheableContainer.hpp>
```

Classes

class CachedImage

A CachedImage is a specialized Image object that exposes the setupDrawChain() method.

Public Member Functions

CacheableContainer ()

Default constructor.

virtual ∼CacheableContainer ()

Destructor.

void setCacheBitmap (BitmapId bitmapId)

Set the dynamic bitmap into which the container content will be rendered.

void updateCache ()

Render the container into the attached dynamic bitmap.

void updateCache (const Rect &rect)

Render the container into the attached dynamic bitmap.

void enableCachedMode (bool enable)

Toggle cached mode on and off.

virtual void invalidateRect (Rect &invalidatedArea) const

Request that a subregion of this drawable is redrawn.

· bool isChildInvalidated () const

Queries the CacheableContainer whether any child widget has been invalidated.

Protected Member Functions

virtual void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement)
 For TouchGFX internal use only.

Additional Inherited Members

7.37.1 Detailed Description

A CacheableContainer is a Drawable that can have child nodes. The z-order of children is determined by the order in which Drawables are added to the container - the Drawable added last will be front-most on the screen. A CacheableContainer can also render its content to a dynamic bitmap that could be used as a texture in subsequent drawing operations.

See also

Container Bitmap

7.37.2 Constructor & Destructor Documentation

```
7.37.2.1 CacheableContainer()
```

```
CacheableContainer ( ) [inline]
```

Default constructor.

7.37.2.2 ∼CacheableContainer()

```
~CacheableContainer ( ) [inline], [virtual]
```

Destructor.

7.37.3 Member Function Documentation

7.37.3.1 enableCachedMode()

```
void enableCachedMode (
          bool enable )
```

Toggle cached mode on and off. The cacheable container behaves as a regular container when cached mode is off.

Parameters

enable	Enable or disable cached mode.
--------	--------------------------------

7.37.3.2 invalidateRect()

Request that a subregion of this drawable is redrawn. Will recursively traverse the tree towards the root, and once reached, issue a draw operation. When this function returns, the specified invalidated area has been redrawn for all appropriate Drawables covering the region.

Parameters

in	invalidatedArea	The area of this drawable to redraw expressed in coordinates relative to its parent (e.g.	
		to request a complete redraw, invalidatedArea will be (0, 0, width, height).	

Reimplemented from Drawable.

7.37.3.3 isChildInvalidated()

```
void isChildInvalidated ( ) const
```

Queries the CacheableContainer whether any child widget has been invalidated.

Returns

True if a child widget has been invalidated and false otherwise.

7.37.3.4 setCacheBitmap()

Set the dynamic bitmap into which the container content will be rendered.

Parameters

bitmap⇔	Id of the dynamic bitmap to serve as a render target.
ld	

7.37.3.5 setupDrawChain()

Configure linked list for draw chain.

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

Reimplemented from Container.

```
7.37.3.6 updateCache() [1/2]
void updateCache ( )
```

Render the container into the attached dynamic bitmap.

Render the container into the attached dynamic bitmap. Only the specified Rect region is updated.

Parameters

rect	Region to update.
------	-------------------

7.38 CacheableContainer::CachedImage Class Reference

A CachedImage is a specialized Image object that exposes the setupDrawChain() method.

```
#include <touchgfx/containers/CacheableContainer.hpp>
```

Public Member Functions

• CachedImage ()

Default constructor.

virtual ∼CachedImage ()

Destructor.

• void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement) For TouchGFX internal use only.

Additional Inherited Members

7.38.1 Detailed Description

A CachedImage is a specialized Image object that exposes the setupDrawChain() method.

See also

CacheableContainer Image

7.38.2 Constructor & Destructor Documentation

```
7.38.2.1 CachedImage()
```

```
CachedImage ( ) [inline]
```

Default constructor.

7.38.2.2 \sim CachedImage()

```
\simCachedImage ( ) [inline], [virtual]
```

Destructor.

7.38.3 Member Function Documentation

7.38.3.1 setupDrawChain()

Configure linked list for draw chain.

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

Reimplemented from Drawable.

7.39 Bitmap::CacheTableEntry Struct Reference

Cache bookkeeping.

#include <touchgfx/Bitmap.hpp>

Public Attributes

• uint8_t * data

Pointer to location of image data for this bitmap in the cache. 0 if bitmap not cached.

7.39.1 Detailed Description

Cache bookkeeping.

7.40 Callback< dest_type, T1, T2, T3 > Struct Template Reference

A Callback is basically a wrapper of a pointer-to-member-function.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

· Callback ()

Default constructor.

• Callback (dest type *pobject, void(dest type::*pmemfun 3)(T1, T2, T3))

Initializes a Callback with an object and a pointer to the member function in that object to call.

• virtual void execute (T1 t1, T2 t2, T3 t3)

Calls the member function.

· virtual bool is Valid () const

Function to check whether the Callback has been initialized with values.

7.40.1 Detailed Description

```
template<class dest_type, typename T1 = void, typename T2 = void, typename T3 = void> struct touchgfx::Callback< dest_type, T1, T2, T3 >
```

A Callback is basically a wrapper of a pointer-to-member-function.

It is used for registering callbacks between widgets. For instance, a Button can be configured to call a member function when it is clicked.

The class is templated in order to provide the class type of the object in which the member function resides, and the argument types of the function to call.

The Callback class exists in four versions, for supporting member functions with 0, 1, 2 or 3 arguments. The compiler will infer which type to use automatically.

Note

The member function to call must return void. The function can have zero, one, two or three arguments of any type.

Template Parameters

-	
dest_type	The type of the class in which the member function resides.
T1	The type of the first argument in the member function, or void if none.
T2	The type of the second argument in the member function, or void if none.
T3	The type of the third argument in the member function, or void if none.

TouchGFX v4.12.3 (C)opyright STMicroelectronics 2014-2019

7.40.2 Constructor & Destructor Documentation

Initializes a Callback with an object and a pointer to the member function in that object to call.

Parameters

in	pobject	Pointer to the object on which the function should be called.
in	pmemfun⇔	Address of member function. This is the version where function takes three arguments.
	_3	

7.40.3 Member Function Documentation

7.40.3.1 execute()

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

t1 This value wil		This value will be passed as the first argument in the function call.
t	2	This value will be passed as the second argument in the function call.
t	3	This value will be passed as the third argument in the function call.

Implements GenericCallback < T1, T2, T3 >.

7.40.3.2 isValid()

```
bool isValid ( ) const [inline], [virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

Implements GenericCallback< T1, T2, T3 >.

7.41 Callback< dest_type, T1, T2, void > Struct Template Reference

A Callback is basically a wrapper of a pointer-to-member-function.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

· Callback ()

Default constructor.

Callback (dest_type *pobject, void(dest_type::*pmemfun_2)(T1, T2))

Initializes a Callback with an object and a pointer to the member function in that object to call.

virtual void execute (T1 t1, T2 t2)

Calls the member function.

virtual bool isValid () const

Function to check whether the Callback has been initialized with values.

7.41.1 Detailed Description

```
template<class dest_type, typename T1, typename T2> struct touchgfx::Callback< dest_type, T1, T2, void >
```

A Callback is basically a wrapper of a pointer-to-member-function.

It is used for registering callbacks between widgets. For instance, a Button can be configured to call a member function when it is clicked.

The class is templated in order to provide the class type of the object in which the member function resides, and the argument types of the function to call.

The Callback class exists in four versions, for supporting member functions with 0, 1, 2 or 3 arguments. The compiler will infer which type to use automatically.

Note

The member function to call must return void. The function can have zero, one, two or three arguments of any type.

Template Parameters

dest_type	The type of the class in which the member function resides.
T1	The type of the first argument in the member function, or void if none.
T2	The type of the second argument in the member function, or void if none.

7.41.2 Constructor & Destructor Documentation

```
7.41.2.1 Callback() [1/2]
```

```
Callback ( ) [inline]
```

Initializes an empty callback.

7.41.2.2 Callback() [2/2]

Initializes a Callback with an object and a pointer to the member function in that object to call.

Parameters

in	pobject	Pointer to the object on which the function should be called.
in	pmemfun⊷	Address of member function. This is the version where function takes two arguments.
	_2	

7.41.3 Member Function Documentation

7.41.3.1 execute()

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

t1	This value will be passed as the first argument in the function call.
t2	This value will be passed as the second argument in the function call.

7.41.3.2 isValid()

```
bool isValid ( ) const [inline], [virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

Implements GenericCallback< T1, T2 >.

7.42 Callback< dest_type, T1, void, void > Struct Template Reference

A Callback is basically a wrapper of a pointer-to-member-function.

#include <touchgfx/Callback.hpp>

Public Member Functions

• Callback ()

Default constructor.

Callback (dest_type *pobject, void(dest_type::*pmemfun_1)(T1))

Initializes a Callback with an object and a pointer to the member function in that object to call.

• virtual void execute (T1 t1)

Calls the member function.

• virtual bool isValid () const

Query if this object is valid.

7.42.1 Detailed Description

```
template < class dest_type, typename T1 > struct touchgfx::Callback < dest_type, T1, void, void >
```

A Callback is basically a wrapper of a pointer-to-member-function.

It is used for registering callbacks between widgets. For instance, a Button can be configured to call a member function when it is clicked.

The class is templated in order to provide the class type of the object in which the member function resides, and the argument types of the function to call.

The Callback class exists in four versions, for supporting member functions with 0, 1, 2 or 3 arguments. The compiler will infer which type to use automatically.

Note

The member function to call must return void. The function can have zero, one, two or three arguments of any type.

Template Parameters

dest_type	The type of the class in which the member function resides.
T1	The type of the first argument in the member function, or void if none.

7.42.2 Constructor & Destructor Documentation

```
void(dest_type::*)(T1) pmemfun_1 ) [inline]
```

Initializes a Callback with an object and a pointer to the member function in that object to call.

Parameters

in	pobject	Pointer to the object on which the function should be called.
in	pmemfun⇔	Address of member function. This is the version where function takes one argument.
	_1	

7.42.3 Member Function Documentation

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

t1 This value will be passed as the first argument in the function call.

See also

isValid()

7.42.3.2 isValid()

```
bool isValid ( ) const [inline], [virtual]
```

Query if this object is valid.

Returns

true if valid, false if not.

Implements GenericCallback< T1 >.

7.43 Callback< dest_type, void, void, void > Struct Template Reference

A Callback is basically a wrapper of a pointer-to-member-function.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

• Callback ()

Default constructor.

Callback (dest_type *pobject, void(dest_type::*pmemfun_0)())

Initializes a Callback with an object and a pointer to the member function in that object to call.

• virtual void execute ()

Calls the member function.

virtual bool isValid () const

Function to check whether the Callback has been initialized with values.

7.43.1 Detailed Description

```
template < class dest_type > struct touchgfx::Callback < dest_type, void, void, void >
```

A Callback is basically a wrapper of a pointer-to-member-function.

It is used for registering callbacks between widgets. For instance, a Button can be configured to call a member function when it is clicked.

The class is templated in order to provide the class type of the object in which the member function resides, and the argument types of the function to call.

The Callback class exists in four versions, for supporting member functions with 0, 1, 2 or 3 arguments. The compiler will infer which type to use automatically.

Note

The member function to call must return void. The function can have zero, one, two or three arguments of any type.

Template Parameters

```
dest_type The type of the class in which the member function resides.
```

7.43.2 Constructor & Destructor Documentation

```
7.43.2.1 Callback() [1/2]

Callback ( ) [inline]

Initializes an empty callback.
```

```
7.43.2.2 Callback() [2/2]
```

Initializes a Callback with an object and a pointer to the member function in that object to call.

Parameters

in	pobject Pointer to the object on which the function should be called.	
in	pmemfun⊷	Address of member function. This is the version where function takes zero arguments.
	_0	

7.43.3 Member Function Documentation

7.43.3.1 execute()

```
void execute ( ) [inline], [virtual]
```

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

7.43.3.2 isValid()

```
bool isValid ( ) const [inline], [virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

Implements GenericCallback<>.

7.44 Keyboard::CallbackArea Struct Reference

Mapping from rectangle to a callback method to execute.

```
#include <touchgfx/widgets/Keyboard.hpp>
```

Public Attributes

· Rect keyArea

The area occupied by a key.

• GenericCallback * callback

The callback to execute, when the area is "pressed". The callback should be a Callback

YourClass> member in the class using the keyboard.

· Bitmapld highlightBitmapld

A bitmap to show when the area is "pressed".

7.45 Canvas Class Reference

Class for easy rendering using CanvasWidgetRenderer.

```
#include <touchgfx/widgets/canvas/Canvas.hpp>
```

Public Member Functions

• Canvas (const CanvasWidget *_widget, const Rect &invalidatedArea)

Canvas Constructor.

virtual ∼Canvas ()

Destructor.

void moveTo (CWRUtil::Q5 x, CWRUtil::Q5 y)

Move the current pen position.

void lineTo (CWRUtil::Q5 x, CWRUtil::Q5 y)

Draw line from current pen position.

```
template<typename T > void moveTo (T x, T y)
```

Move the current pen position.

template<typename T > void lineTo (T x, T y)

Draw line from current pen position.

• bool render ()

Render the drawn shape.

7.45.1 Detailed Description

The Canvas class will make implementation of a new CanvasWidget very easy. The few simple primitives allows moving a "pen" and drawing the outline of a shape which can then be rendered.

The Canvas class has been optimized to eliminate drawing unnecessary lines above and below the currently invalidated rectangle. This was chosen because CanvasWidgetRenderer works with horizontal scan lines, and eliminating unnecessary lines on the left and right does result in as good optimizations, and in some cases (as e.g. Circle) work against the desired (and expected) optimization.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 Canvas()

Canvas Constructor. Locks the frame buffer and prepares for drawing only in the allowed area which has been invalidated. The color depth of the LCD is taken into account.

Parameters

_widget	a pointer to the CanvasWidget using this Canvas. Used for getting the canvas dimensions.
invalidatedArea	the are which should be updated.

```
7.45.2.2 \simCanvas()
```

```
~Canvas ( ) [virtual]
```

Destructor. Takes care of unlocking the frame buffer.

7.45.3 Member Function Documentation

Mark the line from the current (x, y) to the new (x, y) as part of the shape being drawn. As for moveTo, moveTo and lineTo commands completely outside the drawing are are discarded.

Parameters

χ	(The x coordinate for the pen position in Q5 format.
У	/	The y coordinate for the pen position in Q5 format.

See also

```
CWRUtil::Q5 moveTo
```

```
7.45.3.2 lineTo() [2/2] template< typename T > void lineTo ( T x,
```

T y) [inline]

Mark the line from the current (x, y) to the new (x, y) as part of the shape being drawn. As for moveTo, moveTo and lineTo commands completely outside the drawing are are discarded.

Template Parameters

```
T either int or float.
```

Parameters

X	The x coordinate for the pen position.
у	The y coordinate for the pen position.

Move the current pen position to (x, y). If the pen is outside (above or below) the drawing area, nothing is done, but the coordinates are saved in case the next operation is lineTo a coordinate which is inside (or on the opposite side of) the drawing area.

Parameters

Χ	The x coordinate for the pen position in Q5 format.
У	The y coordinate for the pen position in Q5 format.

See also

CWRUtil::Q5 lineTo

7.45.3.4 moveTo() [2/2]

Move the current pen position to (x, y). If the pen is outside (above or below) the drawing area, nothing is done, but the coordinates are saved in case the next operation is line To a coordinate which is inside (or on the opposite side of) the drawing area.

Template Parameters

Parameters

X	The x coordinate for the pen position.
У	The y coordinate for the pen position.

7.45.3.5 render()

```
bool render ( )
```

Render the graphical shape drawn (using moveTo() and lineTo()) using the widgets Painter. The shape is automatically closed, i.e. a lineTo() is automatically inserted connecting the current pen position with the initial pen position given in the previous moveTo() command.

Returns

true if the widget was rendered, false if insufficient memory was available to render the widget.

7.46 CanvasWidget Class Reference

Class for drawing complex polygons on the LCD using CanvasWidgetRenderer.

```
#include <touchgfx/widgets/canvas/CanvasWidget.hpp>
```

Public Member Functions

• CanvasWidget ()

Constructor.

virtual ∼CanvasWidget ()

Destructor

· virtual void setPainter (AbstractPainter &painter)

Sets the painter for the CanvasWidget.

· virtual AbstractPainter & getPainter () const

Gets the current painter for the CanvasWidget.

virtual void setAlpha (uint8_t alpha)

Sets the alpha channel for the CanvasWidget.

• virtual uint8_t getAlpha () const

Gets the current alpha value.

· virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

· virtual void invalidate () const

Invalidates the area covered by this CanvasWidget.

· virtual Rect getMinimalRect () const

Gets minimal rectangle containing the shape drawn by this widget.

virtual Rect getSolidRect () const

Gets the largest solid (non-transparent) rectangle.

virtual bool drawCanvasWidget (const Rect &invalidatedArea) const =0

Draw canvas widget for the given invalidated area.

Additional Inherited Members

7.46.1 Detailed Description

Class for drawing complex polygons on the LCD using CanvasWidgetRenderer.

See also

Widget

7.46.2 Constructor & Destructor Documentation

```
7.46.2.1 CanvasWidget()
```

```
CanvasWidget ( )
```

Constructor.

7.46.2.2 \sim CanvasWidget()

```
~CanvasWidget ( ) [virtual]
```

Destructor. Declared virtual for sub classing purposes.

7.46.3 Member Function Documentation

7.46.3.1 draw()

Draws the given invalidated area. If the underlying CanvasWidgetRenderer fail to render the widget (due to memory limitations), the invalidated area is cut into smaller slices which are then drawn separately. If drawing a single raster line fails, that line is skipped (left blank/transparent) and drawing continues on the next raster line.

If drawing has failed at least once, the number of successfully drawn lines is remembered for the next time. If a future draw would need to draw more lines, the area is automatically divided into smaller areas to prevent drawing the canvas widget in vain.

Parameters

invalidatedArea The invalidated area.

See also

drawCanvasWidget()

Implements Drawable.

7.46.3.2 drawCanvasWidget()

Draw canvas widget for the given invalidated area. Similar to draw(), but might be invoked with several times with smaller areas to due to memory constraints from the underlying CanvasWidgetRenderer.

Parameters

invalidatedArea	The invalidated area.
-----------------	-----------------------

Returns

true the widget was drawn, false if not.

See also

draw()

Implemented in Circle, Line, and AbstractShape.

7.46.3.3 getAlpha()

```
uint8_t getAlpha ( ) const [inline], [virtual]
```

Gets the current alpha value.

Returns

The current alpha value.

7.46.3.4 getMinimalRect()

```
Rect getMinimalRect ( ) const [virtual]
```

Gets minimal rectangle containing the shape drawn by this widget. Default implementation returns the size of the entire widget, but this function should be overwritten in subclasses and return the minimal rectangle containing the shape. See classes such as Circle for example implementations.

Returns

The minimal rectangle containing the shape drawn by this widget.

Reimplemented in Circle, AbstractShape, and Line.

7.46.3.5 getPainter()

```
AbstractPainter & getPainter ( ) const [virtual]
```

Gets the current painter for the CanvasWidget.

Returns

The painter.

See also

AbstractPainter

7.46.3.6 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets the largest solid (non-transparent) rectangle. Since canvas widgets typically do not have a solid rect, it is recommended to return an empty rectangle.

Returns

The largest solid (non-transparent) rectangle.

Implements Drawable.

7.46.3.7 invalidate()

```
void invalidate ( ) const [virtual]
```

Invalidates the area covered by this CanvasWidget. Since many widgets are a lot smaller than the actual size of the canvas widget, each widget must be able to tell the smallest rectangle completely containing the shape drawn by the widget. For example a circle arc is typically much smaller than the widget containing the circle.

See also

getMinimalRect()

Reimplemented from Drawable.

7.46.3.8 setAlpha()

Sets the alpha channel for the CanvasWidget.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.46.3.9 setPainter()

Sets the painter for the CanvasWidget.

Note

The area containing the CanvasWidget is not invalidated.

Parameters

	in	painter	The painter for the CanvasWidget.	
--	----	---------	-----------------------------------	--

See also

AbstractPainter

7.47 CanvasWidgetRenderer Class Reference

Class for supporting drawing of figures.

```
#include <touchgfx/canvas_widget_renderer/CanvasWidgetRenderer.hpp>
```

Static Public Member Functions

• static void setupBuffer (uint8_t *buffer, unsigned bufsize)

Setup the buffers used by CanvasWidget.

• static bool setScanlineWidth (unsigned width)

Sets scanline width.

• static bool hasBuffer ()

Query if CanvasWidgetRenderer has been initialized with a buffer.

• static unsigned getScanlineWidth ()

The width of a scanline.

• static void * getScanlineCovers ()

Gets pointer to memory used for covers in Scanline.

• static void * getScanlineStartIndices ()

Gets pointer to memory used for indices in Scanline.

static void * getScanlineCounts ()

Gets pointer to memory used for counts in Scanline.

static Cell * getOutlineBuffer ()

Gets pointer to memory used for Cell objects in Outline.

static unsigned int getOutlineBufferSize ()

Gets size of memory area used for Cell objects in Outline.

static void setWriteMemoryUsageReport (bool writeUsageReport)

Memory reporting.

static bool getWriteMemoryUsageReport ()

Gets write memory usage report flag.

• static void numCellsUsed (unsigned used)

Called by the destructor in Outline to help keep track of the memory requirements of CanvasWidgets.

static void numCellsMissing (unsigned missing)

Called by the destructor in Outline to help keep track of the memory requirements of CanvasWidgets.

• static unsigned getUsedBufferSize ()

Calculate how much memory has been required by CanvasWidgets.

· static unsigned getMissingBufferSize ()

Calculate how much memory was required by CanvasWidgets, but was unavailable.

Additional Inherited Members

7.47.1 Detailed Description

Class for supporting drawing of figures. This class holds the memory which is used by the underlying algorithms. CanvasWidget will not allocate memory dynamically, but will use memory from the buffer kept in CanvasWidget Renderer. When using the TouchGFX simulator, it is also possible to get a report on the actual amount of memory used for the drawings to help adjusting the buffer size.

See also

Widget

7.47.2 Member Function Documentation

7.47.2.1 getMissingBufferSize()

```
static unsigned getMissingBufferSize ( ) [static]
```

Calculate how much memory was required by CanvasWidgets, but was unavailable. If the value returned is greater than 0 it means the This can be used to fine tune the size of the buffer passed to CanvasWidgetRenderer upon initialization.

Returns

The number of bytes required.

7.47.2.2 getOutlineBuffer()

```
static Cell * getOutlineBuffer ( ) [static]
```

Gets pointer to memory used for Cell objects in Outline.

Returns

Pointer to memory used internally by Outline.

7.47.2.3 getOutlineBufferSize()

```
static unsigned int getOutlineBufferSize ( ) [static]
```

Gets size of memory area used for Cell objects in Outline.

Returns

Size of memory area used internally by Outline.

7.47.2.4 getScanlineCounts()

```
static void * getScanlineCounts ( ) [static]
```

Gets pointer to memory used for counts in Scanline.

Returns

Pointer to memory used internally by Scanline.

7.47.2.5 getScanlineCovers()

```
static void * getScanlineCovers ( ) [static]
```

Gets pointer to memory used for covers in Scanline.

Returns

Pointer to memory used internally by Scanline.

7.47.2.6 getScanlineStartIndices()

```
static void * getScanlineStartIndices ( ) [static]
```

Gets pointer to memory used for indices in Scanline.

Returns

Pointer to memory used internally by Scanline.

7.47.2.7 getScanlineWidth()

```
static unsigned getScanlineWidth ( ) [static]
```

The width of a scanline. This is the same as the width of the invalidated area. Used to optimize the memory layout of the buffer.

Returns

Scanline width (HAL::FRAME_BUFFER_WIDTH).

7.47.2.8 getUsedBufferSize()

```
static unsigned getUsedBufferSize ( ) [static]
```

Calculate how much memory has been required by CanvasWidgets. This can be used to fine tune the size of the buffer passed to CanvasWidgetRenderer upon initialization.

Returns

The number of bytes required.

7.47.2.9 getWriteMemoryUsageReport()

```
static bool getWriteMemoryUsageReport ( ) [static]
```

Gets write memory usage report flag.

Returns

true if it CWR writes memory reports, false if not.

7.47.2.10 hasBuffer()

```
static bool hasBuffer ( ) [static]
```

Query if CanvasWidgetRenderer has been initialized with a buffer.

Returns

True if a buffer has been setup.

7.47.2.11 numCellsMissing()

Called by the destructor in Outline to help keep track of the memory requirements of CanvasWidgets.

Parameters

missing Number of Cell objects required, but not available, to Outline.

7.47.2.12 numCellsUsed()

```
static void numCellsUsed ( unsigned \ used \ ) \quad [static] \\
```

Called by the destructor in Outline to help keep track of the memory requirements of CanvasWidgets.

Parameters

used Number of Cell objects used from the dedicated buffer.

7.47.2.13 setScanlineWidth()

Sets scanline width. Setting the scanline width will initialize the buffers for scanline and outline. If the width set is too large to fit the scanline buffers in the allocated memory buffer, false will be returned and all buffer pointers will be cleared.

Parameters

width	The width of the scanline on screen.
-------	--------------------------------------

Returns

true if it succeeds, false if it fails.

7.47.2.14 setupBuffer()

Setup the buffers used by CanvasWidget.

Parameters

in	buffer	Buffer reserved for CanvasWidget.
	bufsize	The size of the buffer.

7.47.2.15 setWriteMemoryUsageReport()

Memory reporting can be turned on (and off) using this method. CWR will try to work with the given amount of memory passed when calling setupBuffer(). If the outline of the figure is too complex, this will be reported.

"CWR requires X bytes" means that X bytes is the highest number of bytesd requied by CWR so far, but since the size of the invalidated area and the shape of things draw can influence this, this may be reported several times with

a higher and higher number. Leave your app running for a long time to find out what the memory requirements are.

"CWR requires X bytes (Y bytes missing)" means the same as the report above, but there as was not enough memory to render the entire shape. To get around this, CWR will split the shape into two separate drawings of half size. This means that less memory is required, but drawing will be (somewhat) slower. After you see this message all future draws will be split into smaller chunks, so memory requirements might not get as high. This is followed by:

"CWR will split draw into multiple draws due to limited memory." actually just means that CWR will try to work with a smaller amount of memory.

In general, if there is enough memory available to run the simulation and never see the message "CWR will split draw ...", this is preferred. The size of the buffer required will be the highest number X reported as "CWR requires X bytes". Good numbers can also be around half of X.

Parameters

writeUsageReport	true to write report.
------------------	-----------------------

See also

setupBuffer

7.48 Cell Struct Reference

A pixel cell.

```
#include <touchgfx/canvas_widget_renderer/Cell.hpp>
```

Public Member Functions

void set (int _x, int _y, int _cover, int _area)

Sets all the Cell parameters.

void setCoord (int _x, int _y)

Sets the coordinate of the Cell.

void setCover (int cover, int area)

Sets the cover of area cell.

void addCover (int _cover, int _area)

Adds a cover to a Cell.

• int packedCoord () const

Packed coordinates of the Cell.

Public Attributes

int16_t x

The x coordinate.

int16_t y

The y coordinate.

• int16 t cover

The cover (see http://projects.tuxee.net/cl-vectors/section-the-cl-aa-algorithm for further information).

• int16 tarea

The area (see http://projects.tuxee.net/cl-vectors/section-the-cl-aa-algorithm for further information).

7.48 Cell Struct Reference 221

7.48.1 Detailed Description

A pixel cell. There are no constructors defined and it was done intentionally in order to avoid extra overhead when allocating an array of cells.

7.48.2 Member Function Documentation

7.48.2.1 addCover()

Adds a cover to a Cell.

Parameters

_cover	The cover to add to the Cell.
_area	The area to add to the Cell.

7.48.2.2 packedCoord()

```
int packedCoord ( ) const [inline]
```

Packed coordinates of the Cell. By packing the x coordinate and y coordinate into one int, it is possible to sort Cells using a single comparison.

Returns

The packed coordinates with y in the high part and x in the low part.

7.48.2.3 set()

Sets all the Cell parameters.

Parameters

_X	The x coordinate.
_y	The y coordinate.
_cover	The cover.
_area	The area.

7.48.2.4 setCoord()

Sets the coordinate of the Cell.

Parameters

\leftarrow	The Cell's x coordinate.
_←	
X	
\leftrightarrow	The Cell's y coordinate.
_←	
У	

7.48.2.5 setCover()

Sets the cover of area cell.

Parameters

_cover	The cover.
_area	The area.

7.49 Circle Class Reference

Simple widget capable of drawing a circle.

```
#include <touchgfx/widgets/canvas/Circle.hpp>
```

Public Member Functions

• Circle ()

Constructs a new Circle.

template<typename T > void setCircle (const T x, const T y, const T r)

Sets the center and radius of the Circle.

void setCircle (const int16_t x, const int16_t y, const int16_t r)

Sets the center and radius of the Circle.

template<typename T >
 void setCenter (const T x, const T y)

Sets the center of the Circle.

• void setCenter (const int16_t x, const int16_t y)

Sets the center of the Circle.

template<typename T >
 void getCenter (T &x, T &y) const

• template<typename T >

Gets the center coordinates of the Circle.

void setRadius (const T r) Sets the radius of the Circle. template<typename T > void getRadius (T &r) const Gets the radius of the Circle. • template<typename T > void setArc (const T startAngle, const T endAngle) Sets the start and end angles in degrees of the Circle arc. void setArc (const int16 t startAngle, const int16 t endAngle) Sets the start and end angles in degrees of the Circle arc. • template<typename T > void getArc (T &startAngle, T &endAngle) const Gets the start and end angles in degrees for the circle arc. int16 t getArcStart () const Gets the start angle in degrees for the arc. • template<typename T > void getArcStart (T &angle) const Gets the start angle in degrees for the arc. • int16_t getArcEnd () const Gets the end angle in degrees for the arc. template<typename T > void getArcEnd (T &angle) const Gets the end angle in degrees for the arc. template<typename T > void updateArcStart (const T startAngle) Updates the start angle in degrees for this Circle arc. • template<typename T > void updateArcEnd (const T endAngle) Updates the end angle in degrees for this Circle arc. template<typename T > void updateArc (const T startAngle, const T endAngle) Updates the start and end angle in degrees for this Circle arc. template<typename T > void setLineWidth (const T width) Sets the line width for this Circle. template<typename T > void getLineWidth (T &width) const Gets line width. void setPrecision (const int precision) Sets a precision of the Circle drawing function. • int getPrecision () const Gets the precision of the circle drawing function. void setCapPrecision (const int precision) Sets the precision of the ends of the Circle arc. • int getCapPrecision () const Sets the precision of the ends of the Circle arc. virtual bool drawCanvasWidget (const Rect &invalidatedArea) const Draws the Circle. · virtual Rect getMinimalRect () const Gets minimal rectangle for the current shape of the circle.

Rect getMinimalRect (int16_t arcStart, int16_t arcEnd) const

Gets minimal rectangle containing a given circle arc.

• Rect getMinimalRect (CWRUtil::Q5 arcStart, CWRUtil::Q5 arcEnd) const

Gets minimal rectangle containing a given circle arc.

Additional Inherited Members

7.49.1 Detailed Description

Simple widget capable of drawing a circle. By tweaking the parameters of the circle, several parameters of the circle can be changed. Center, radius, line width, line cap and partial circle arc. This opens for creation of fascinating graphics.

See also

CanvasWidget

7.49.2 Constructor & Destructor Documentation

7.49.2.1 Circle()

```
Circle ()
```

Constructs a new Circle.

7.49.3 Member Function Documentation

7.49.3.1 drawCanvasWidget()

Draws the Circle. This class supports partial drawing, so only the area described by the rectangle will be drawn.

Parameters

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.
-----------------	--

Returns

true if it succeeds, false if it fails.

Implements CanvasWidget.

7.49.3.2 getArc()

7.49 Circle Class Reference 225

Gets the start and end angles in degrees for the circle arc.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

out	startAngle	The start.
out	endAngle	The end.

See also

setArc

```
7.49.3.3 getArcEnd() [1/2] template< typename T > T getArcEnd ( ) const [inline]
```

Gets the end angle in degrees for the arc.

Template Parameters

```
T Generic type parameter, either int (default) or float.
```

Returns

The finishing angle for the arc.

See also

getArc setArc

```
7.49.3.4 getArcEnd() [2/2]
```

```
template< typename T > void getArcEnd ( T & angle ) const [inline]
```

Gets the end angle in degrees for the arc.

Template Parameters

T Generic type parameter, either int or float.

Parameters

in,out	angle	The angle.

```
7.49.3.5 getArcStart() [1/2]
int16_t getArcStart ( ) const [inline]
```

Gets the start angle in degrees for the arc.

Returns

The starting angle for the arc.

See also

getArc setArc

7.49.3.6 getArcStart() [2/2]

Gets the start angle in degrees for the arc.

Template Parameters

T Generic type parameter, either int or float.

Parameters

in,out	angle	The angle.
--------	-------	------------

See also

getArc setArc

7.49.3.7 getCapPrecision()

```
int getCapPrecision ( ) const
```

Gets the precision of the ends of the $\mbox{\it Circle}$ arc.

Returns

The cap precision in degrees.

See also

getCapPrecision

7.49 Circle Class Reference 227

7.49.3.8 getCenter()

```
template< typename T > void getCenter ( T & x, T & y ) const [inline]
```

Gets the center coordinates of the Circle.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

out	Х	The x coordinate of the center.
out	у	The y coordinate of the center.

7.49.3.9 getLineWidth()

```
template< typename T > void getLineWidth ( T & width ) const [inline]
```

Gets line width.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

out	width	The width.

See also

setLineWidth

```
7.49.3.10 getMinimalRect() [1/3]
```

```
Rect getMinimalRect ( ) const [virtual]
```

Gets minimal rectangle for the current shape of the circle.

Returns

The minimal rectangle.

Reimplemented from CanvasWidget.

7.49.3.11 getMinimalRect() [2/3]

```
Rect getMinimalRect (
                int16_t arcStart,
                int16_t arcEnd ) const
```

Gets minimal rectangle containing a given circle arc.

Parameters

arcStart	The arc start.
arcEnd	The arc end.

Returns

The minimal rectangle.

7.49.3.12 getMinimalRect() [3/3]

Gets minimal rectangle containing a given circle arc.

Parameters

arcStart	The arc start.
arcEnd	The arc end.

Returns

The minimal rectangle.

7.49.3.13 getPrecision()

```
int getPrecision ( ) const
```

Gets the precision of the circle drawing function. The precision is the number of degrees used as step counter when drawing smaller line fragments around the circumference of the circle, the default being being 5.

Returns

The precision.

See also

setPrecision

7.49 Circle Class Reference 229

7.49.3.14 getRadius()

```
template< typename T > void getRadius ( T & r ) const [inline]
```

Gets the radius of the Circle.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

```
out r The radius.
```

7.49.3.15 setArc() [1/2]

Sets the start and end angles in degrees of the Circle arc. 0 degrees is straight up (12 o'clock) and 90 degrees is to the left (3 o'clock). Any positive or negative degrees can be used to specify the part of the Circle to draw.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

startAngle	The start degrees.
endAngle	The end degrees.

Note

The area containing the Circle is not invalidated.

See also

```
getArc
updateArcStart
updateArcEnd
updateArc
```

7.49.3.16 setArc() [2/2]

Sets the start and end angles in degrees of the Circle arc. 0 degrees is straight up (12 o'clock) and 90 degrees is to the left (3 o'clock). Any positive or negative degrees can be used to specify the part of the Circle to draw.

Parameters

startAngle	The start degrees.
endAngle	The end degrees.

Note

The area containing the Circle is not invalidated.

See also

```
getArc
updateArcStart
updateArcEnd
updateArc
```

7.49.3.17 setCapPrecision()

```
\begin{tabular}{ll} \beg
```

Sets a precision of the ends of the Circle arc. The precision is given in degrees where 180 is the default which results in a square ended arc (aka "butt cap"). 90 will draw "an arrow head" and smaller values gives a round cap. Larger values of precision results in faster rendering of the circle.

Parameters

precision	The new cap precision.
-----------	------------------------

Note

The circle is not invalidated.

7.49.3.18 setCenter() [1/2]

Sets the center of the Circle.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

Χ	The x coordinate of center.
У	The y coordinate of center.

Note

The area containing the Circle is not invalidated.

```
7.49.3.19 setCenter() [2/2] void setCenter ( const int16_t x, const int16_t y) [inline]
```

Sets the center of the Circle.

Parameters

Х	The x coordinate of center.
У	The y coordinate of center.

Note

The area containing the Circle is not invalidated.

Sets the center and radius of the Circle.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

Х	The x coordinate of center.
у	The y coordinate of center.
r	The radius.

Note

The area containing the Circle is not invalidated.

Sets the center and radius of the Circle.

Parameters

Χ	The x coordinate of center.
у	The y coordinate of center.
r	The radius.

Note

The area containing the Circle is not invalidated.

7.49.3.22 setLineWidth()

Sets the line width for this Circle. If the line width is set to zero, the circle will be filled.

Template Parameters

T Generic type parameter, either int or flo	at.
---	-----

Parameters

th The width of the line measured in pixels	 3.
---	--------

Note

The area containing the Circle is not invalidated.

if the new line with is smaller than the old width, the circle should be invalidated before updating the width to ensure that the old circle is completely erased.

7.49.3.23 setPrecision()

Sets a precision of the Circle drawing function. The number given as precision is the number of degrees used as step counter when drawing smaller line fragments around the circumference of the circle, five being a sensible value. Higher values results in less nice circles but faster rendering. Large circles might need a precision smaller than five.

Parameters

precision	The precision measured in degrees.

Note

The circle is not invalidated.

7.49 Circle Class Reference 233

7.49.3.24 setRadius()

```
template< typename T > void setRadius ( const T r ) [inline]
```

Sets the radius of the Circle.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

```
r The radius.
```

Note

The area containing the Circle is not invalidated.

7.49.3.25 updateArc()

```
template< typename T > void updateArc ( const T startAngle, const T endAngle) [inline]
```

Updates the start and end angle in degrees for this Circle arc.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

startAngle	The new start angle in degrees.
endAngle	The new end angle in degrees.

Note

The areas containing the updated Circle arcs are invalidated.

See also

setArc getArc updateArcStart updateArcEnd

7.49.3.26 updateArcEnd()

Updates the end angle in degrees for this Circle arc.

Template Parameters

T Generic type parameter, either int or float.

Parameters

endAngle	The end angle in degrees.
----------	---------------------------

Note

The area containing the updated Circle arc is invalidated.

See also

```
setArc
updateArcStart
updateArc
```

7.49.3.27 updateArcStart()

```
\label{template} \mbox{typename T} > \mbox{void updateArcStart (} \\ \mbox{const T } \mbox{startAngle )} \quad \mbox{[inline]}
```

Updates the start angle in degrees for this Circle arc.

Template Parameters

```
T Generic type parameter, either int or float.
```

Parameters

Note

The area containing the updated Circle arc is invalidated.

See also

```
setArc
updateArcEnd
updateArc
```

7.50 CircleProgress Class Reference

A circle progress.

```
#include <touchgfx/containers/progress_indicators/CircleProgress.hpp>
```

Public Member Functions

CircleProgress ()

Default constructor.

virtual ∼CircleProgress ()

Destructor.

• virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the position and dimension of the circle progress indicator.

• virtual void setPainter (AbstractPainter &painter)

Sets the painter.

virtual void setCenter (int x, int y)

Sets the center.

• virtual void getCenter (int &x, int &y) const

Gets the center coordinates.

virtual void setRadius (int r)

Sets the radius.

· virtual int getRadius () const

Gets the radius.

virtual void setLineWidth (int width)

Sets line width.

• virtual int getLineWidth () const

Gets line width.

• virtual void setCapPrecision (int precision)

Sets the cap precision.

• virtual int getCapPrecision () const

Gets the cap precision.

• virtual void setStartEndAngle (int startAngle, int endAngle)

Sets start and end angle.

• virtual int getStartAngle () const

Gets start angle.

• virtual int getEndAngle () const

Gets end angle.

virtual void setAlpha (uint8_t alpha)

Sets the alpha.

• virtual uint8_t getAlpha () const

Gets the alpha.

virtual void setValue (int value)

Sets a value.

Protected Attributes

· Circle circle

The circle.

· int circleEndAngle

The circle end angle.

Additional Inherited Members

7.50.1 Detailed Description

A circle progress indicator uses CWR for drawing the arc of a circle to show progress, and so the user must create a painter for painting the circle. The circle progress is defined by setting the minimum and maximum angle of the arc.

See also

Circle

7.50.2 Constructor & Destructor Documentation

```
7.50.2.1 CircleProgress()
CircleProgress ( )
Default constructor.

7.50.2.2 ~CircleProgress()
~CircleProgress ( ) [virtual]
Destructor.
```

7.50.3 Member Function Documentation

```
7.50.3.1 getAlpha()

uint8_t getAlpha ( ) const [virtual]

Gets the alpha of the circle.
```

Returns

The alpha.

See also

setAlpha

```
7.50.3.2 getCapPrecision()
```

```
int getCapPrecision ( ) const [inline], [virtual]
```

Gets the cap precision.

Returns

The cap precision.

7.50.3.3 getCenter()

Gets the center coordinates.

Parameters

out	X	The x coordinate.
out	у	The y coordinate.

7.50.3.4 getEndAngle()

```
int getEndAngle ( ) const [virtual]
```

Gets end angle. Beware that the value returned is not related to the current progress of the circle but rather the end point of the circle when it is at 100%.

Returns

The end angle.

See also

```
setStartEndAngle
setEndAngle
```

7.50.3.5 getLineWidth()

```
int getLineWidth ( ) const [virtual]
```

Gets line width.

Returns

The line width.

See also

setLineWidth

```
7.50.3.6 getRadius()
```

```
int getRadius ( ) const [virtual]
```

Gets the radius.

Returns

The radius.

7.50.3.7 getStartAngle()

```
int getStartAngle ( ) const [virtual]
```

Gets start angle.

Returns

The start angle.

See also

```
setStartEndAngle
getEndAngle
```

7.50.3.8 setAlpha()

Sets the alpha of the Circle. Please note, that the alpha can also be set on the Painter, but this can be controlled directly from the user app, setting alpha for the CircleProgress will set the alpha of the actual circle.

Parameters

```
alpha The alpha.
```

See also

getAlpha

7.50.3.9 setCapPrecision()

Sets the cap precision of the circle arc. This is not used if line width is zero.

Parameters

```
precision The cap precision.
```

See also

Circle::setCapPrecision

7.50.3.10 setCenter()

Sets the center of the circle / arc.

Parameters

X	The int to process.
У	The int to process.

7.50.3.11 setLineWidth()

Sets line width of the circle. If a line width of zero is specified, it has a special meaning of drawing a filled circle instead of just the circle arc.

Parameters

See also

Circle::setLineWidth

7.50.3.12 setPainter()

Sets the painter to use for drawing.

Parameters

in, out painter The painter.

See also

Circle::setPainter AbstractPainter

7.50.3.13 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the circle progress indicator relative to the background image.

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the circle progress indicator.
height	The height of the circle progress indicator.

Reimplemented from AbstractProgressIndicator.

7.50.3.14 setRadius()

```
void setRadius (
          int r ) [virtual]
```

Sets the radius of the circle.

Parameters

r The int to process.	
-----------------------	--

See also

Circle::setRadius

7.50.3.15 setStartEndAngle()

Sets start and end angle. By swapping end and start angles, circles can be drawn backwards.

Parameters

startAngle	The start angle.
endAngle	The end angle.

7.50.3.16 setValue()

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

value	The value.

Reimplemented from AbstractProgressIndicator.

7.51 ClickButtonTrigger Class Reference

A click button trigger.

#include <touchgfx/containers/buttons/ClickButtonTrigger.hpp>

Public Member Functions

• ClickButtonTrigger ()

Default constructor.

virtual ~ClickButtonTrigger ()

Destructor.

virtual void handleClickEvent (const ClickEvent &event)

Handles the click event described by event.

Additional Inherited Members

7.51.1 Detailed Description

A click button trigger. This trigger will create a button that reacts on clicks. This means it will call the action when it gets a touch released event.

The ClickButtonTrigger can be combined with one or more of the ButtonStyle classes to create a functional button.

7.51.2 Member Function Documentation

7.51.2.1 handleClickEvent()

Handles the click event described by event. The action callback is called when receiving a REALEASED event in PRESSED state.

Parameters

```
event The event.
```

Reimplemented from Drawable.

7.52 ClickEvent Class Reference

A click event.

```
#include <touchgfx/events/ClickEvent.hpp>
```

Public Types

enum ClickEventType { PRESSED, RELEASED, CANCEL }

The click event types.

Public Member Functions

ClickEvent (ClickEventType type, int16_t x, int16_t y, int16_t force=0)

Constructor.

virtual ∼ClickEvent ()

Destructor.

int16_t getX () const

Gets the x coordinate of this event.

• int16_t getY () const

Gets the y coordinate of this event.

void setX (int16_t x)

Sets the x coordinate of this event.

void setY (int16_t y)

Sets the y coordinate of this event.

void setType (ClickEventType type)

Sets the click type of this event.

ClickEventType getType () const

Gets the click type of this event.

• int16_t getForce () const

Gets the force of the click.

virtual Event::EventType getEventType ()

Gets event type.

7.52.1 Detailed Description

A click event. The semantics of this event is slightly depending on hardware platform. ClickEvents are generated by the HAL layer.

See also

Event

7.52.2 Member Enumeration Documentation

7.52.2.1 ClickEventType

enum enum ClickEventType

Enumerator

PRESSED	An enum constant representing the pressed option.
RELEASED	An enum constant representing the released option.
CANCEL	An enum constant representing the cancel option.

7.52.3 Constructor & Destructor Documentation

7.52.3.1 ClickEvent()

Constructor.

Parameters

type	The type of the click event.	
X	The x coordinate of the click event.	
У	The y coordinate of the click event.	
force	orce The force of the click. On touch displays this usually means how hard the user pressed on the display	
	On the windows platform, this will always be zero.	

```
~ClickEvent ( ) [inline], [virtual]
```

Destructor.

7.52.4 Member Function Documentation

7.52.4.1 getEventType()

```
Event::EventType getEventType ( ) [inline], [virtual]
```

Gets event type.

Returns

The type of this event.

Implements Event.

7.52.4.2 getForce()

```
int16_t getForce ( ) const [inline]
```

Gets the force of the click. On touch displays this usually means how hard the user pressed on the display. On the windows platform, this will always be zero.

Returns

The force of the click.

7.52.4.3 getType()

```
ClickEventType getType ( ) const [inline]
```

Gets the click type of this event.

Returns

The click type of this event.

7.52.4.4 getX()

```
int16_t getX ( ) const [inline]
```

Gets the x coordinate of this event.

Returns

The x coordinate of this event.

7.52.4.5 getY()

```
int16_t getY ( ) const [inline]
```

Gets the y coordinate of this event.

Returns

The y coordinate of this event.

7.52.4.6 setType()

Sets the click type of this event.

Parameters

tvpe	The type to set.
ιγρο	The type to set.

7.52.4.7 setX()

```
void setX (
          int16_t x ) [inline]
```

Sets the x coordinate of this event.

Parameters 4 8 1

	l -
V	The x coordinate of this event

7.52.4.8 setY()

```
void setY (
                     int16_t y ) [inline]
```

Sets the y coordinate of this event.

Parameters

y The y coordinate of this event.

7.53 ClickListener < T > Class Template Reference

Mix-in class that extends a class with a click action event.

#include <touchgfx/mixins/ClickListener.hpp>

Public Member Functions

· ClickListener ()

Default constructor.

virtual ∼ClickListener ()

Destructor.

· virtual void handleClickEvent (const ClickEvent &event)

Ensures that the clickEvent is propagated to the super class.

• void setClickAction (GenericCallback< const T &, const ClickEvent & > &callback)

Associates an action to be performed when the class T is clicked.

Protected Attributes

GenericCallback< const T &, const ClickEvent & > * clickAction

The callback to be executed when T is clicked.

7.53.1 Detailed Description

```
template < class T > class touchgfx::ClickListener < T >
```

Mix-in class that extends a class with a click action event that is called when the class receives a click event.

Template Parameters

T | specifies the type to extend with the ClickListener behavior.

7.53.2 Constructor & Destructor Documentation

7.54 Color Class Reference 247

7.53.2.1 ClickListener()

```
ClickListener ( ) [inline]
```

Default constructor.

7.53.3 Member Function Documentation

7.53.3.1 handleClickEvent()

Ensures that the clickEvent is propagated to the super class T and to the clickAction listener.

Parameters

```
event Information about the click.
```

See also

Drawable::handleClickEvent()

7.53.3.2 setClickAction()

Associates an action to be performed when the class T is clicked.

Parameters

callback	The callback to be executed.	The callback will be given a reference to T	<u>г. </u>
----------	------------------------------	---	---

See also

GenericCallback

7.54 Color Class Reference

Contains functionality for color conversion.

```
#include <touchgfx/Color.hpp>
```

Static Public Member Functions

- static colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue)
 Generates a color representation to be used on the LCD, based on 24 bit RGB values. Depending on your chosen color bit depth, the color will be interpreted internally as either a 16 bit or 24 bit color value.
- static uint8_t getRedColor (colortype color)

Gets the red color part of a color.

static uint8_t getGreenColor (colortype color)

Gets the green color part of a color.

• static uint8_t getBlueColor (colortype color)

Gets the blue color part of a color.

7.54.1 Detailed Description

Contains functionality for color conversion.

7.54.2 Member Function Documentation

7.54.2.1 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The 16 bit color value.
-------	-------------------------

Returns

The blue part of the color.

7.54.2.2 getColorFrom24BitRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values. Depending on your chosen color bit depth, the color will be interpreted internally as either a 16 bit or 24 bit color value. This function can be safely used regardless of whether your application is configured for 16 or 24 bit colors.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.54.2.3 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

7.54.2.4 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The color value.
```

Returns

The red part of the color.

7.55 colortype Struct Reference

Type for representing a color.

```
#include <touchgfx/hal/Types.hpp>
```

Public Member Functions

• colortype ()

Default constructor.

• colortype (uint32_t col)

Constructor.

• uint32_t getColor32 () const

Gets color 32 as a 32bit value suitable for passing to Color::getRedColor(), Color::getGreenColor() and Color::get← BlueColor() which will handle both 16 bit colortypes and 24 bit colortypes.

• operator uint16_t () const

Cast that converts the given colortype to an uint16_t.

Public Attributes

uint32_t color
 The color.

7.55.1 Detailed Description

This type can contain a color value. Note that in order to maintain backwards compatibility, casting this type to an integral value will yield a 16-bit value. To extract a 24/32-bit color from this type, use the getColor32 function.

7.55.2 Constructor & Destructor Documentation

Contructor which creates a colortype with the given color. Use Color::getColorFrom24BitRGB() to create a color that will work on both 16 bit LCD and 24 bit LCD.

Parameters

```
col The color.
```

See also

Color::getColorFrom24BitRGB()

7.55.3 Member Function Documentation

```
7.55.3.1 getColor32()
uint32_t getColor32 ( ) const [inline]
Returns
```

See also

Color::getRedColor()
Color::getGreenColor()
Color::getBlueColor()

The color 32.

7.55.3.2 operator uint16_t()

```
operator uint16_t ( ) const [inline]
```

Cast that converts the given colortype to an uint16_t. Provided only for backward compatibility. Not recommended to use.

Returns

The result of the operation.

7.56 ConstFont Class Reference

A ConstFont is a Font implementation that has its contents defined at compile-time and usually placed in read-only memory.

```
#include <touchgfx/ConstFont.hpp>
```

Public Member Functions

• ConstFont (const GlyphNode *list, uint16_t size, uint16_t height, uint8_t pixBelowBase, uint8_t bitsPerPixel, uint8_t dataFormatA4, uint8_t maxLeft, uint8_t maxRight, const Unicode::UnicodeChar fallbackChar, const Unicode::UnicodeChar ellipsisChar)

Creates a font instance.

virtual const GlyphNode * getGlyph (Unicode::UnicodeChar unicode, const uint8_t *&pixelData, uint8_←
t &bitsPerPixel) const

Gets the glyph data associated with the specified unicode.

virtual const uint8 t * getPixelData (const GlyphNode *glyph) const =0

Gets the pixel date associated with this glyph.

• virtual int8_t getKerning (Unicode::UnicodeChar prevChar, const GlyphNode *glyph) const =0

Gets the kerning distance between two characters.

const GlyphNode * find (Unicode::UnicodeChar unicode) const

Finds the glyph data associated with the specified unicode.

Protected Attributes

· const GlyphNode * glyphList

The list of glyphs.

• uint16 t listSize

The size of the list of glyphs.

Additional Inherited Members

7.56.1 Detailed Description

A ConstFont is a Font implementation that has its contents defined at compile-time and usually placed in read-only memory.

Note

Pure virtual class. Create an application- specific implementation of getPixelData.

See also

Font

7.56.2 Constructor & Destructor Documentation

7.56.2.1 ConstFont()

Creates a font instance.

Parameters

list	The array of glyphs known to this font.
size	The number of glyphs in list.
height	The height in pixels of the highest character in this font.
pixBelowBase	The maximum number of pixels that can be drawn below the baseline in this font.
bitsPerPixel	The number of bits per pixel in this font.
dataFormatA4	The glyphs are saved using ST A4 format.
maxLeft	The maximum a character extends to the left.
maxRight	The maximum a character extends to the right.
fallbackChar	The fallback character for the typography in case no glyph is available.
ellipsisChar	The ellipsis character used for truncating long texts.

7.56.3 Member Function Documentation

```
7.56.3.1 find()
```

Finds the glyph data associated with the specified unicode.

Parameters

unicode	The character to look up.
---------	---------------------------

Returns

A pointer to the glyph node or null if the glyph was not found.

7.56.3.2 getGlyph()

Gets the glyph data associated with the specified unicode. An implementation of Font::getGlyph. Searches the glyph list for the specified font.

Complexity O(log n)

Parameters

	unicode	The character to look up.
	pixelData	Pointer to the pixel data for the glyph if the glyph is found. This is set by this method.
out	bitsPerPixel	Reference where to place the number of bits per pixel.

Returns

A pointer to the glyph node or null if the glyph was not found.

Implements Font.

7.56.3.3 getKerning()

Gets the kerning distance between two characters.

Parameters

prevChar	The unicode value of the previous character.
glyph	the glyph object for the current character.

Returns

The kerning distance between prevChar and glyph char.

Reimplemented from Font.

Implemented in InternalFlashFont.

7.56.3.4 getPixelData()

Gets the pixel date associated with this glyph.

Parameters

glyph	The glyph to get the pixels data from.

Returns

Pointer to the pixel data of this glyph.

Implemented in InternalFlashFont.

7.57 Container Class Reference

A Container is a Drawable that can have child nodes.

#include <touchgfx/containers/Container.hpp>

Public Member Functions

· Container ()

Default constructor.

virtual ∼Container ()

Destructor.

virtual void add (Drawable &d)

Adds a Drawable instance as child to this Container.

• virtual void remove (Drawable &d)

Removes a Drawable instance from the list of children.

virtual void removeAll ()

Removes all children by resetting their parent and sibling pointers.

virtual void unlink ()

Removes all children by unlinking the first child.

virtual bool contains (const Drawable &d)

Query if this object contains the given drawable.

virtual void insert (Drawable *previous, Drawable &d)

Inserts a Drawable instance after the specified child node.

virtual void getLastChild (int16_t x, int16_t y, Drawable **last)

Gets the last child of this container.

virtual void draw (const Rect &invalidatedArea) const

Draws the children of this container.

virtual Rect getSolidRect () const

Gets solid rectangle of a Container.

virtual void forEachChild (GenericCallback
 Drawable &> *function)

Calls the specified function for each child in the container.

virtual uint16_t getType () const

For GUI testing only.

Drawable * getFirstChild ()

Obtain a pointer to the first child of this container.

Protected Member Functions

· virtual Rect getContainedArea () const

Gets a rectangle describing the total area covered by the children of this container.

virtual void moveChildrenRelative (int16_t deltaX, int16_t deltaY)

Calls moveRelative on all children.

virtual void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement)

For TouchGFX internal use only.

Protected Attributes

Drawable * firstChild

Pointer to the first child of this container. Subsequent children can be found through firstChild->nextSibling.

Additional Inherited Members

7.57.1 Detailed Description

A Container is a Drawable that can have child nodes. The z-order of children is determined by the order in which Drawables are added to the container - the Drawable added last will be front-most on the screen.

This class overrides a few functions in Drawable in order to traverse child nodes.

Note that containers act as view ports - that is, only the parts of children that intersect with the geometry of the container will be visible (e.g. setting a container's width to 0 will render all children invisible).

See also

Drawable

7.57.2 Constructor & Destructor Documentation

```
7.57.2.1 Container()
Container ( ) [inline]
Default constructor.
7.57.2.2 ~Container()
~Container ( ) [inline], [virtual]
Destructor.
```

7.57.3 Member Function Documentation

Adds a Drawable instance as child to this Container.

Note

Never add a drawable more than once (will loop forever)!

Parameters

ı	in	d	The Drawable to add.
	T11	u	THE DIAWADIE ID AUG.

Reimplemented in SlideMenu, ScrollableContainer, ModalWindow, ListLayout, and SwipeContainer.

7.57.3.2 contains()

```
bool contains (  {\tt const\ Drawable\ \&\ d\ )} \quad [{\tt virtual}]
```

Query if this object contains the given drawable.

Parameters

```
d The Drawable to look for.
```

Returns

True if the specified Drawable instance is direct child of this container, false otherwise.

7.57.3.3 draw()

Draws the children of this container. Only used when JSMOC is disabled.

Parameters

- 4		
ſ	invalidatedArea	The area to draw.
	IIIvaliualeuAlea	THE area to draw.

Implements Drawable.

Reimplemented in Keyboard.

7.57.3.4 forEachChild()

Calls the specified function for each child in the container. Function to call must have the following prototype: void T::func(Drawable&)

Parameters

in	function	The function to be executed for each child.

See also

ListLayout::insert for a usage example.

7.57.3.5 getContainedArea()

```
Rect getContainedArea ( ) const [protected], [virtual]
```

Gets a rectangle describing the total area covered by the children of this container.

Returns

Rectangle covering all children.

Reimplemented in ScrollableContainer.

7.57.3.6 getFirstChild()

```
Drawable * getFirstChild ( ) [inline]
```

Useful if you want to manually iterate the children added to this container.

Returns

Pointer to the first drawable added to this container. If nothing has been added return zero.

7.57.3.7 getLastChild()

```
void getLastChild (
    int16_t x,
    int16_t y,
    Drawable ** last ) [virtual]
```

Gets the last (=highest Z-order) child of this container that is enabled, visible and intersects with the specified point. Recursive function.

Parameters

	X	The x coordinate of the intersection.
	У	The y coordinate of the intersection.
out	last	out parameter in which the result is placed.

See also

Drawable::getLastChild

Implements Drawable.

Reimplemented in ScrollableContainer.

7.57.3.8 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets solid rectangle of a Container. JSMOC does not operate directly on containers.

Returns

An empty rectangle per default.

Implements Drawable.

```
7.57.3.9 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_CONTAINER.

Reimplemented from Drawable.

Reimplemented in ZoomAnimationImage, ScrollableContainer, Slider, Keyboard, and ListLayout.

7.57.3.10 insert()

Inserts a Drawable instance after the specified child node. If previous is null, the drawable will be inserted as the first element in the list.

Note

As with add, do not add the same drawable twice.

Parameters

in	previous	The Drawable to insert after. If null, insert as header.
in	d	The Drawable to insert.

Reimplemented in ListLayout.

7.57.3.11 moveChildrenRelative()

```
void moveChildrenRelative (
          int16_t deltaX,
          int16_t deltaY) [protected], [virtual]
```

Calls moveRelative on all children.

Parameters

deltaX	Horizontal displacement.
deltaY	Vertical displacement.

Reimplemented in ScrollableContainer.

7.57.3.12 remove()

Removes a Drawable instance from the list of children.

Note

This is safe to call even if d is not a child (in which case nothing happens).

Parameters

in d The Drawable to remove.

Reimplemented in SlideMenu, ModalWindow, ListLayout, and SwipeContainer.

7.57.3.13 removeAll()

```
void removeAll ( ) [virtual]
```

Removes all children by resetting their parent and sibling pointers.

Reimplemented in ListLayout.

7.57.3.14 setupDrawChain()

Configure linked list for draw chain.

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

Reimplemented from Drawable.

Reimplemented in Keyboard, and CacheableContainer.

7.57.3.15 unlink()

```
void unlink ( ) [virtual]
```

Removes all children by unlinking the first child. The parent and sibling pointers of the children are not reset.

7.58 CoverTransition < templateDirection > Class Template Reference

A Transition that slides from one screen to the next.

```
#include <touchgfx/transitions/CoverTransition.hpp>
```

Classes

class FullSolidRect

A Widget that returns a solid rect of the same size as the application.

Public Member Functions

• CoverTransition (const uint8 t transitionSteps=20)

Constructor.

virtual ∼CoverTransition ()

Destructor.

virtual void handleTickEvent ()

Handles the tick event when transitioning.

virtual void tearDown ()

Tear down.

· virtual void init ()

Initializes this object.

Protected Member Functions

• virtual void initMoveDrawable (Drawable &d)

Moves the Drawable to its initial position.

virtual void tickMoveDrawable (Drawable &d)

Moves the Drawable.

Additional Inherited Members

7.58.1 Detailed Description

```
template<Direction templateDirection> class touchgfx::CoverTransition< templateDirection>
```

A Transition that slides the new screen over the previous.

Template Parameters

templateDirection	Type of the template direction.
-------------------	---------------------------------

See also

Transition

7.58.2 Constructor & Destructor Documentation

7.58.2.1 CoverTransition()

Constructor.

Parameters

transitionSteps Number of steps in the transition animation.

7.58.2.2 ∼CoverTransition()

```
\simCoverTransition ( ) [inline], [virtual]
```

Destructor.

7.58.3 Member Function Documentation

7.58.3.1 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

Handles the tick event when transitioning. It moves the contents of the Screen's container. The direction of the transition determines the direction the contents of the container moves.

Reimplemented from Transition.

7.58.3.2 init()

```
void init ( ) [inline], [virtual]
```

Initializes this object.

See also

Transition::init()

Reimplemented from Transition.

7.58.3.3 initMoveDrawable()

Moves the Drawable to its initial position outside of the visible area.

Parameters

```
in d The Drawable to move.
```

7.58.3.4 tearDown()

```
void tearDown ( ) [inline], [virtual]
```

Tear down.

Transition::tearDown()

Reimplemented from Transition.

7.58.3.5 tickMoveDrawable()

Moves the Drawable.

Parameters

in d The Drawable	to move.
-------------------	----------

7.59 CWRUtil Struct Reference

Helper classes and functions for CanvasWidget.

```
#include <touchgfx/widgets/canvas/CWRUtil.hpp>
```

Classes

• class Q10

Defines a number with 10 bits reserved for fraction.

• class Q15

Defines a number with 15 bits reserved for fraction.

• class Q5

Defines a number with 5 bits reserved for fraction.

Static Public Member Functions

```
    static FORCE_INLINE_FUNCTION Q5 toQ5 (Q5 value)

      Convert a Q5 to itself.
• template<typename T >
  static FORCE INLINE FUNCTION Q5 toQ5 (T value)
      Convert an integer to a fixed point number.
• template<typename T >
  static FORCE_INLINE_FUNCTION Q10 toQ10 (T value)
      Convert an integer to a fixed point number.

    static Q15 sine (int i)

      Find the value of sin(i) with 15 bits precision.
• static Q15 sine (Q5 i)
      Find the value of sin(i) with 15 bits precision.

    static Q15 cosine (int i)

      Find the value of cos(i) with 15 bits precision.

    static Q15 cosine (Q5 i)

      Find the value of cos(i) with 15 bits precision.

    static int8_t arcsine (Q10 q10)

      Gets the arcsine of the given fraction (given as Q10).
• template<typename T >
  static int angle (T x, T y)
      Find angle of a coordinate.
template<typename T >
  static int angle (T x, T y, T &d)
      Find angle of a coordinate.
• static int angle (Q5 x, Q5 y)
      Find angle of a coordinate.

    static int angle (Q5 x, Q5 y, Q5 &d)

      Find the angle of the coordinate (x, y).

    static Q5 sqrtQ10 (Q10 value)

      Find the square root of the given value.

    static Q5 muldivQ5 (Q5 factor1, Q5 factor2, Q5 divisor)

      Multiply two Q5's and divide by a Q5 without overflowing the multiplication.

    static Q5 muldivQ10 (Q10 factor1, Q10 factor2, Q10 divisor)

      Multiply two Q5's and divide by a Q5 without overflowing the multiplication.

    static Q5 mulQ5 (Q5 factor1, Q5 factor2)

      Multiply two Q5's returning a new Q5.

    static Q5 mulQ5 (Q5 factor1, Q10 factor2)

      Multiply one Q5 by a Q10 returning a new Q5.
```

7.59.1 Detailed Description

Helper classes and functions for CanvasWidget. A handful of utility functions can be found here. These include helper functions for converting between float, int and Q5/Q10/Q15 format. There are also functions for calculating sin() and cos() in integers with a high number of bits reserved for fraction. Having sin() and cos() pre- calculated in this way allows very fast drawing of circles without the need for floating point arithmetic.

Using Q5, numbers from -1024.00000 to +1024.96875 with a precision of 1/32 = 0.03125 can be represented.

See also

```
http://en.wikipedia.org/wiki/Q_%28number_format%29
Widget
```

7.59.2 Member Function Documentation

Template Parameters

Τ	Generic type parameter (int or float).
---	--

Parameters

X	The x coordinate.
у	The y coordinate.

Returns

The angle of the coordinate.

Template Parameters

Parameters

	X	The x coordinate.
	У	The y coordinate.
out	d	The distance from (0,0) to (x,y).

Returns

The angle of the coordinate.

Parameters

Х	The x coordinate.
У	The y coordinate.

Returns

The angle of the coordinate.

```
7.59.2.4 angle() [4/4] static int angle ( 25 x, 25 y, 25 \& d) [inline], [static]
```

Parameters

	Х	The x coordinate.
	у	The y coordinate.
out	d	The distance from (0,0) to (x,y).

Returns

The angle.

7.59.2.5 arcsine()

```
static int8_t arcsine ( $\tt Q10\ q10\ )$ [inline], [static]
```

Gets the arcsine of the given fraction (given as Q10). The function is most precise for angles 0-45. To limit memory requirements, values above sqrt(1/2) is calculated as 90-arcsine($sqrt(1-q10^{4})$). Internally.

Parameters

```
q10 The 10.
```

Returns

An int8_t.

```
7.59.2.6 \operatorname{cosine}() [1/2] static Q15 \operatorname{cosine}( int i ) [inline], [static]
```

Find the value of cos(i) with 15 bits precision using the fact that cos(i)=sin(90-i).

Parameters

i the angle in degrees. The angle follows the angles of the clock, 0 being straight up and 90 being 3 o'clock.

Returns

the value of cos(i) with 15 bits precision on the fractional part.

See also

sine()

```
7.59.2.7 cosine() [2/2] static Q15 cosine ( Q5 i ) [inline], [static]
```

Find the value of cos(i) with 15 bits precision using the fact that cos(i)=sin(90-i).

Parameters

i the angle in degrees. The angle follows the angles of the clock, 0 being straight up and 90 being 3 o'clock.

Returns

the value of cos(i) with 15 bits precision on the fractional part.

See also

sine()

7.59.2.8 muldivQ10()

```
static Q5 muldivQ10 (
          Q10 factor1,
          Q10 factor2,
          Q10 divisor ) [inline], [static]
```

Multiply two Q5's and divide by a Q5 without overflowing the multiplication (assuming that the final result can be stored in a Q5).

Parameters

factor1	The first factor.
factor2	The second factor.
divisor	The divisor.

Returns

factor1 * factor2 / divisor.

7.59.2.9 muldivQ5()

```
static Q5 muldivQ5 (
        Q5 factor1,
        Q5 factor2,
        Q5 divisor ) [inline], [static]
```

Multiply two Q5's and divide by a Q5 without overflowing the multiplication (assuming that the final result can be stored in a Q5).

Parameters

factor1	The first factor.
factor2	The second factor.
divisor	The divisor.

Returns

factor1 * factor2 / divisor.

Multiply two Q5's returning a new Q5 without overflowing.

Parameters

factor1	The first factor.
factor2	The second factor.

Returns

factor1 * factor2.

Multiply one Q5 by a Q10 returning a new Q5 without overflowing.

Parameters

factor1	The first factor.
factor2	The second factor.

Returns

factor1 * factor2.

```
7.59.2.12 sine() [1/2] static Q15 sine( int i) [inline], [static]
```

Find the value of sin(i) with 15 bits precision. The returned value can be converted to a floating point number and divided by (1<<15) to get the rounded value of sin(i). By using this function, a complete circle can be drawn without the need for using floating point math.

Parameters

i the angle in degrees. The angle follows the angles of the clock, 0 being straight up and 90 being 3 o'clock.

Returns

the value of sin(i) with 15 bits precision on the fractional part.

```
7.59.2.13 sine() [2/2]
static Q15 sine (
          Q5 i ) [inline], [static]
```

Find the value of $\sin(i)$ with 15 bits precision. The returned value can be converted to a floating point number and divided by (1<<15) to get the rounded value of $\sin(i)$. By using this function, a complete circle can be drawn without the need for using floating point math.

If the given degree is not an integer, the value is approximated by interpolation between sin(floor(i)) and sin(ceil(i)).

Parameters

i the angle in degrees. The angle follows the angles of the clock, 0 being straight up and 90 being 3 o'clock.

Returns

the value of sin(i) with 15 bits precision on the fractional part.

7.59.2.14 sqrtQ10()

Parameters

value The value to find the square root of.

Returns

The square root of the given value.

7.59.2.15 toQ10()

```
template< typename T > FORCE_INLINE_FUNCTION static Q10 toQ10 ( T value ) [inline], [static]
```

Convert an integer to a fixed point number. This is done by shifting the integer value 10 places to the left, or multiplying the floating point value by (1 < < 510).

Template Parameters

T Should be either int or float.

Parameters

value	the integer to convert.
-------	-------------------------

Returns

the converted integer.

Convert a Q5 to itself. Allows toQ5 to be called with a variable that is already Q5.

Parameters

```
value the Q5.
```

Returns

the value passed.

Convert an integer to a fixed point number. This is done by shifting the integer value 5 places to the left, or multiplying the floating point value by (1 << 5)

Template Parameters

T Should be either int or float.

Parameters

value	the integer to convert.
value	the integer to convert.

Returns

the converted integer.

7.60 DebugPrinter Class Reference

The class DebugPrinter defines the interface for printing debug messages on top of the framebuffer.

```
#include <touchqfx/lcd/LCD.hpp>
```

Public Member Functions

void setDebugString (const char *string)

Sets the debug string to be displayed on top of the framebuffer.

void setDebugPosition (uint16_t x, uint16_t y, uint16_t w, uint16_t h)

Sets the position of the debug string.

void setDebugScale (uint8_t scale)

Sets the font scale of the debug string.

void setDebugColor (colortype fg)

Sets the foreground color of the debug string.

virtual void draw (const LCD &lcd) const =0

Draws the debug string on top of the framebuffer content.

• const Rect & region () const

Returns the region of the debug string.

Protected Attributes

const char * debugString

Debug string to be displayed onscreen.

· Rect debugRegion

Region onscreen where the debug message is displayed.

• colortype debugForegroundColor

Font color to use when displaying the debug string.

uint8_t debugScale

Font scaling factor to use when displaying the debug string.

7.60.1 Detailed Description

The class DebugPrinter defines the interface for printing debug messages on top of the framebuffer.

7.60.2 Member Function Documentation

```
7.60.2.1 draw()  \begin{tabular}{ll} \begin{tabular}{ll} void draw ( & & lcd ) const [pure virtual] \end{tabular}
```

Draws the debug string on top of the framebuffer content.

Parameters

in	lcd	Reference on the LCD object to use for drawing the debug string.
----	-----	--

Implemented in LCD24DebugPrinter.

7.60.2.2 region()

```
const Rect & region ( ) const [inline]
```

Returns the region where the debug string is displayed.

Returns

Rect The debug string region.

7.60.2.3 setDebugColor()

Sets the foreground color of the debug string.

Parameters

iı	l	fg	The foreground color of the debug string.
----	---	----	---

7.60.2.4 setDebugPosition()

Sets the position onscreen where the debug string will be displayed.

Parameters

ir	ì	Χ	The coordinate of the region where the debug string is displayed.
ir	1	y	The coordinate of the region where the debug string is displayed.
ir	1	W	The width of the region where the debug string is displayed.
ir	1	h	The height of the region where the debug string is displayed.

7.60.2.5 setDebugScale()

Sets the font scale of the debug string.

Parameters

in	scale	The font scale of the debug string.
----	-------	-------------------------------------

7.60.2.6 setDebugString()

Sets the debug string to be displayed on top of the framebuffer.

Parameters

	in	string	The string to be displayed.	
--	----	--------	-----------------------------	--

7.61 DigitalClock Class Reference

A digital clock.

#include <touchgfx/containers/clock/DigitalClock.hpp>

Public Types

enum DisplayMode { DISPLAY_12_HOUR_NO_SECONDS, DISPLAY_24_HOUR_NO_SECONDS, DISP
 LAY_12_HOUR, DISPLAY_24_HOUR }

Values that represent different display modes.

Public Member Functions

• DigitalClock ()

Default constructor.

virtual ∼DigitalClock ()

Destructor.

virtual void setWidth (int16_t width)

Sets the width of the DigitalClock.

virtual void setHeight (int16_t height)

Sets the height of the DigitalClock.

virtual void setBaselineY (int16_t baselineY)

Adjusts the digital clocks y coordinate to place the text at the specified baseline.

virtual void setTypedText (TypedText typedText)

Sets the typed text of the DigitalClock.

virtual void setColor (colortype color)

Sets the color of the text.

· virtual void setDisplayMode (DisplayMode dm)

Sets the display mode.

· virtual DisplayMode getDisplayMode () const

Gets the display mode.

void displayLeadingZeroForHourIndicator (bool displayLeadingZero)

Sets whether to display a leading zero for the hour indicator or not.

virtual void setAlpha (uint8 t alpha)

Sets the alpha value of the digital clock.

virtual uint8_t getAlpha () const

Gets the alpha value of the digital clock.

virtual uint16_t getTextWidth () const

Gets text width.

Protected Member Functions

virtual void updateClock ()

Updates the visual representation of the clock.

Protected Attributes

DisplayMode displayMode

The current display mode.

· bool useLeadingZeroForHourIndicator

Print a leading zero if the hour is less than 10.

• TextAreaWithOneWildcard text

The clock text.

• Unicode::UnicodeChar buffer [BUFFER_SIZE]

Wild card buffer for the clock text.

Static Protected Attributes

• static const int BUFFER_SIZE = 16

Buffer size of the wild card.

7.61.1 Detailed Description

A digital clock. Can be set in either 12 or 24 hour mode. Seconds are optional. Width and height must be set manually to match the typography and alignment specified in the text database. The Digital Clock requires a typedText with one wildcard and uses the following characters (not including quotes) "AMP:0123456789" These must be present in the text database with the same typography as the wildcard text. Leading zero for the hour indicator can be enabled/disable by the displayLeadingZeroForHourIndicator method.

7.61.2 Member Enumeration Documentation

7.61.2.1 DisplayMode

enum DisplayMode

Enumerator

DISPLAY_12_HOUR_NO_SECONDS	12 Hour clock. Seconds are not displayed
DISPLAY_24_HOUR_NO_SECONDS	24 Hour clock. Seconds are not displayed
DISPLAY_12_HOUR	12 Hour clock. Seconds are displayed
DISPLAY_24_HOUR	24 Hour clock. Seconds are displayed

7.61.3 Constructor & Destructor Documentation

```
7.61.3.1 DigitalClock()
```

DigitalClock ()

Default constructor.

7.61.3.2 ~DigitalClock()

```
~DigitalClock ( ) [virtual]
```

Destructor.

7.61.4 Member Function Documentation

7.61.4.1 displayLeadingZeroForHourIndicator()

```
void displayLeadingZeroForHourIndicator (
          bool displayLeadingZero )
```

Sets whether to display a leading zero for the hour indicator or not. That is the if an hour value less than 10 will be displayed as "8:" or "08:".

Default value for this setting is false.

Parameters

displayl eadingZero	true = show leading zero. false = do not show leading zero.
alopia Loading Loi o	and - onon loading zoro. Idioo - do not onon loading zoro.

7.61.4.2 getAlpha()

```
uint8_t getAlpha ( ) const [virtual]
```

Gets the alpha value of the digital clock.

Returns

The alpha value. 255 = completely solid. 0 = invisible.

7.61.4.3 getDisplayMode()

```
DisplayMode getDisplayMode ( ) const [inline], [virtual]
```

Returns

The display mode.

7.61.4.4 getTextWidth()

```
uint16_t getTextWidth ( ) const [inline], [virtual]
```

Returns

The text width of the current content of the digital clock.

7.61.4.5 setAlpha()

Sets the alpha value of the digital clock.

Parameters

```
alpha The alpha value. 255 = completely solid. 0 = invisible.
```

7.61.4.6 setBaselineY()

Adjusts the digital clocks y coordinate so the text will have its baseline at the specified value. The placements is relative to the specified TypedText so if this changes you have to set the baseline again.

Note that setTypedText must be called prior to setting the baseline.

Parameters

```
baselineY The y coordinate of the baseline.
```

7.61.4.7 setColor()

Sets the color of the text. Invalidates the DigitalClock.

Parameters

color	The new text color.
COIOI	THE HEW LEXT COIDS.

7.61.4.8 setDisplayMode()

Parameters

dm The new DisplayMode.

7.61.4.9 setHeight()

```
void setHeight (
                int16_t height ) [virtual]
```

Sets the height of the DigitalClock. The text area that displays the clock is expanded to match the dimension of the DigitalClock.

Parameters

```
height The height.
```

Reimplemented from Drawable.

7.61.4.10 setTypedText()

Sets the typed text of the DigitalClock. Expects a typed text with one wildcard and that the following characters are defined in the text spreadsheet (for the same typography):

AMP:0123456789

Invalidates the DigitalClock.

Parameters

```
typedText Describes the typed text to use.
```

7.61.4.11 setWidth()

Sets the width of the DigitalClock. The text area that displays the clock is expanded to match the dimension of the

DigitalClock.

Parameters

Reimplemented from Drawable.

7.61.4.12 updateClock()

```
virtual void updateClock ( ) [protected], [virtual]
```

Updates the visual representation of the clock.

Implements AbstractClock.

7.62 DisplayTransformation Class Reference

Defines transformations from display space to frame buffer space.

#include <touchgfx/transforms/DisplayTransformation.hpp>

Static Public Member Functions

static void transformDisplayToFrameBuffer (int16_t &x, int16_t &y)

Transform x,y from display to frame buffer coordinates.

static void transformDisplayToFrameBuffer (float &x, float &y)

Transform x,y from display to frame buffer coordinates.

• static void transformFrameBufferToDisplay (int16_t &x, int16_t &y)

Transform x,y from frame buffer to display coordinates.

• static void transformDisplayToFrameBuffer (int16_t &x, int16_t &y, const Rect &in)

Transform x,y from coordinates relative to the in rect to frame buffer coordinates.

• static void transformDisplayToFrameBuffer (float &x, float &y, const Rect &in)

Transform x,y from coordinates relative to the in rect to frame buffer coordinates.

static void transformDisplayToFrameBuffer (Rect &r)

Transform rectangle from display to frame buffer coordinates.

• static void transformDisplayToFrameBuffer (Rect &r, const Rect &in)

Transform rectangle r from coordinates relative to the in rect to frame buffer coordinates.

7.62.1 Detailed Description

Defines transformations from display space to frame buffer space. The display might be (considered) in portrait mode from 0,0 to 272,480, while the actual frame buffer is from 0,0 to 480,272. This class handles the transformations.

7.62.2 Member Function Documentation

7.62.2.1 transformDisplayToFrameBuffer() [1/6]

```
static void transformDisplayToFrameBuffer ( int16\_t ~\&~ x, \\ int16\_t ~\&~ y~) ~~ [static]
```

Transform x,y from display to frame buffer coordinates.

Parameters

in,out	Х	the x part to translate.
in,out	У	the y part to translate.

$\textbf{7.62.2.2} \quad transform \textbf{DisplayToFrameBuffer()} \ \ \texttt{[2/6]}$

```
static void transformDisplayToFrameBuffer (  \mbox{float \& } x, \\ \mbox{float \& } y \;) \quad [\mbox{static}]
```

Transform x,y from display to frame buffer coordinates.

Parameters

in,out	X	the x part to translate.
in,out	у	the y part to translate.

7.62.2.3 transformDisplayToFrameBuffer() [3/6]

```
static void transformDisplayToFrameBuffer ( int16\_t ~\&~ x, \\ int16\_t ~\&~ y, \\ const ~Rect ~\&~ in~) ~ [static]
```

Transform x,y from coordinates relative to the in rect to frame buffer coordinates.

Parameters

in,out	Χ	the x part to translate.
in,out	у	the y part to translate.
	in	the rectangle defining the coordinate space.

7.62.2.4 transformDisplayToFrameBuffer() [4/6]

Transform x,y from coordinates relative to the in rect to frame buffer coordinates.

Parameters

in,out	Х	the x part to translate.	
in,out	У	the y part to translate.	
	in	the rectangle defining the coordinate space.	

7.62.2.5 transformDisplayToFrameBuffer() [5/6]

Transform rectangle from display to frame buffer coordinates.

Parameters

in,out	r	the rectangle to translate.
--------	---	-----------------------------

7.62.2.6 transformDisplayToFrameBuffer() [6/6]

Transform rectangle r from coordinates relative to the in rect to frame buffer coordinates.

Parameters

in,out	r	the rectangle to translate.	
	in	the rectangle defining the coordinate space.	

7.62.2.7 transformFrameBufferToDisplay()

Transform x,y from frame buffer to display coordinates.

Parameters

in,out	X	the x part to translate.
in,out	у	the y part to translate.

7.63 DMA_Interface Class Reference

DMA_Interface provides basic functionality and structure for processing "blit" operations using DMA.

#include <touchgfx/hal/DMA.hpp>

Public Member Functions

virtual BlitOperations getBlitCaps ()=0

Gets the blit capabilities of this DMA.

virtual void addToQueue (const BlitOp &op)

Inserts a BlitOp for processing.

virtual void flush ()

This function blocks until all DMA transfers in the queue have been completed.

• virtual void initialize ()

Perform initialization.

• bool isDMARunning ()

Query if the DMA is running.

void setAllowed (bool allowed)

Sets whether or not a DMA operation is allowed to begin.

bool getAllowed () const

Gets whether a DMA operation is allowed to begin.

· virtual void start ()

Signals that DMA transfers can start.

virtual void signalDMAInterrupt ()=0

This function is called automatically by the framework when a DMA interrupt has been received.

uint8_t isDmaQueueEmpty ()

Query if the DMA queue is empty.

• uint8_t isDmaQueueFull ()

Query if the DMA queue is full.

virtual DMAType getDMAType (void)

Function for obtaining the DMA type of the concrete DMA_Interface implementation.

virtual ~DMA_Interface ()

Destructor.

Protected Member Functions

• DMA_Interface (DMA_Queue &dmaQueue)

Constructs a DMA Interface object.

• virtual void execute ()

Performs a queued blit-op.

virtual void executeCompleted ()

To be called when blit-op has been performed.

virtual void seedExecution ()

Called when elements are added to the DMA-queue.

virtual void setupDataCopy (const BlitOp &blitOp)=0

Configures blit-op hardware for a 2D copy as specified by blitOp.

virtual void setupDataFill (const BlitOp &blitOp)=0

Configures blit-op hardware for a 2D fill as specified by blitOp.

virtual void enableAlpha (uint8_t alpha)

Configures blit-op hardware for alpha-blending.

• virtual void disableAlpha ()

Configures blit-op hardware for solid operation (no alpha-blending).

virtual void enableCopyWithTransparentPixels (uint8_t alpha)

Configures blit-op hardware for alpha-blending while simultaneously skipping transparent pixels.

virtual void waitForFrameBufferSemaphore ()

Waits until frame buffer semaphore is available.

Protected Attributes

DMA_Queue & queue

Reference to the DMA queue.

bool isRunning

true if a DMA transfer is currently ongoing.

volatile bool isAllowed

true if DMA transfers are currently allowed.

7.63.1 Detailed Description

DMA_Interface provides basic functionality and structure for processing "blit" operations using DMA.

7.63.2 Constructor & Destructor Documentation

```
7.63.2.1 \sim DMA_Interface()
```

```
\sim\!\!\text{DMA\_Interface} ( ) [inline], [virtual]
```

Destructor.

7.63.2.2 DMA_Interface()

Constructs a DMA Interface object.

Parameters

in	dmaQueue	Reference to the queue of DMA operations.

7.63.3 Member Function Documentation

7.63.3.1 addToQueue()

Inserts a BlitOp for processing. This also potentially starts the DMA controller, if not already running.

Parameters

```
op The operation to add.
```

7.63.3.2 disableAlpha()

```
void disableAlpha ( ) [protected], [virtual]
```

Configures blit-op hardware for solid operation (no alpha-blending)

7.63.3.3 enableAlpha()

Configures blit-op hardware for alpha-blending.

Parameters

-blending value to apply.	alpha	
---------------------------	-------	--

7.63.3.4 enableCopyWithTransparentPixels()

```
void enableCopyWithTransparentPixels (
          uint8_t alpha ) [protected], [virtual]
```

Configures blit-op hardware for alpha-blending while simultaneously skipping transparent pixels.

Parameters

Ipha The alpha-blending value to apply.

7.63.3.5 execute()

```
void execute ( ) [protected], [virtual]
```

Performs a queued blit-op.

7.63.3.6 executeCompleted()

```
void executeCompleted ( ) [protected], [virtual]
```

To be called when blit-op has been performed.

7.63.3.7 flush()

```
void flush ( ) [inline], [virtual]
```

This function blocks until all DMA transfers in the queue have been completed.

Reimplemented in NoDMA.

7.63.3.8 getAllowed()

```
bool getAllowed ( ) const [inline]
```

Gets whether a DMA operation is allowed to begin. Used in single-buffering to avoid changing the frame buffer while display is being updated.

Returns

true if DMA is allowed to start, false if not.

7.63.3.9 getBlitCaps()

```
BlitOperations getBlitCaps ( ) [pure virtual]
```

Gets the blit capabilities of this DMA.

Returns

The blit operations supported by this DMA implementation.

Implemented in NoDMA.

7.63.3.10 getDMAType()

Function for obtaining the DMA type of the concrete DMA_Interface implementation. As default, will return DMA_
TYPE_GENERIC type value.

Returns

a DMAType value of the concrete DMA_Interface implementation.

7.63.3.11 initialize()

```
void initialize ( ) [inline], [virtual]
```

Perform initialization. Does nothing in this base class.

7.63.3.12 isDmaQueueEmpty()

```
uint8_t isDmaQueueEmpty ( )
```

Query if the DMA queue is empty.

Returns

1 if DMA queue is empty, else 0.

7.63.3.13 isDmaQueueFull()

```
uint8_t isDmaQueueFull ( )
```

Query if the DMA queue is full.

Returns

1 if DMA queue is full, else 0.

7.63.3.14 isDMARunning()

```
bool isDMARunning ( ) [inline]
```

Query if the DMA is running.

Returns

true if a DMA operation is currently in progress.

7.63.3.15 seedExecution()

```
void seedExecution ( ) [protected], [virtual]
```

Called when elements are added to the DMA-queue.

Note

The frame buffer must be locked before this method returns if the DMA-queue is non- empty.

7.63.3.16 setAllowed()

Sets whether or not a DMA operation is allowed to begin. Used in single-buffering to avoid changing the frame buffer while display is being updated.

Parameters

```
allowed true if DMA transfers are allowed.
```

7.63.3.17 setupDataCopy()

Configures blit-op hardware for a 2D copy as specified by blitOp.

Parameters

blitOp	The operation to execute.
--------	---------------------------

Implemented in NoDMA.

7.63.3.18 setupDataFill()

Configures blit-op hardware for a 2D fill as specified by blitOp.

Parameters

blitOp	The operation to execute.
--------	---------------------------

Implemented in NoDMA.

7.63.3.19 signalDMAInterrupt()

```
void signalDMAInterrupt ( ) [pure virtual]
```

This function is called automatically by the framework when a DMA interrupt has been received.

Implemented in NoDMA.

7.63.3.20 start()

```
void start ( ) [virtual]
```

Signals that DMA transfers can start. If any elements are in the queue, start it.

7.63.3.21 waitForFrameBufferSemaphore()

```
void waitForFrameBufferSemaphore ( ) [protected], [virtual]
```

Waits until frame buffer semaphore is available (i.e. neither DMA or application is accessing the frame buffer).

7.64 DMA_Queue Class Reference

This class provides an interface for a FIFO (circular) list used by DMA_Interface and descendants for storing Blit← Op's.

```
#include <touchgfx/hal/DMA.hpp>
```

Public Member Functions

virtual bool isEmpty ()=0

Query if this object is empty.

```
• virtual bool isFull ()=0
```

Query if this object is full.

virtual void pushCopyOf (const BlitOp &op)=0

Adds the specified blitop to the queue.

virtual ~DMA_Queue ()

Destructor.

Protected Member Functions

• DMA_Queue ()

Default constructor.

virtual void pop ()=0

Pops an element from the queue.

virtual const BlitOp * first ()=0

Gets the first element in the queue.

7.64.1 Detailed Description

This class provides an interface for a FIFO (circular) list used by DMA_Interface and descendants for storing Blit←Op's.

7.64.2 Constructor & Destructor Documentation

7.64.3 Member Function Documentation

```
7.64.3.1 first()
const BlitOp * first ( ) [protected], [pure virtual]
```

Gets the first element in the queue.

Returns

The first element in the queue.

Implemented in LockFreeDMA_Queue.

```
7.64.3.2 isEmpty()
bool isEmpty ( ) [pure virtual]
Query if this object is empty.
Returns
     true if the queue is empty.
Implemented in LockFreeDMA_Queue.
7.64.3.3 isFull()
bool isFull ( ) [pure virtual]
Query if this object is full.
Returns
     true if the queue is full.
Implemented in LockFreeDMA_Queue.
7.64.3.4 pop()
void pop ( ) [protected], [pure virtual]
Pops an element from the queue.
Implemented in LockFreeDMA_Queue.
7.64.3.5 pushCopyOf()
void pushCopyOf (
              const BlitOp & op ) [pure virtual]
Adds the specified blitop to the queue.
```

Implemented in LockFreeDMA_Queue.

The blitop to add.

7.65 DragEvent Class Reference

A drag event.

Parameters

#include <touchgfx/events/DragEvent.hpp>

Public Types

enum DragEventType { DRAGGED }

The drag event types.

Public Member Functions

DragEvent (DragEventType type, int16 t oldX, int16 t oldY, int16 t newX, int16 t newY)

Constructor.

virtual ∼DragEvent ()

Destructor.

int16_t getOldX () const

Gets the x coordinate where the drag operation was started (dragged from).

int16_t getOldY () const

Gets the y coordinate where the drag operation was started (dragged from).

int16_t getNewX () const

Gets the new x coordinate (dragged to).

int16_t getNewY () const

Gets the new x coordinate (dragged to).

• DragEventType getType () const

Gets the type of this drag event.

• int16_t getDeltaX () const

Gets the distance in x coordinates (how long was the drag).

int16_t getDeltaY () const

Gets the distance in y coordinates (how long was the drag).

virtual Event::EventType getEventType ()

Gets event type.

7.65.1 Detailed Description

A drag event. The only drag event currently supported is DRAGGED, which will be issued every time the input system detects a drag.

See also

Event

7.65.2 Constructor & Destructor Documentation

7.65.2.1 DragEvent()

Constructor. Create a drag event of the specified type with the specified coordinates.

Parameters

type	The type of the drag event.
oldX	The x coordinate of the drag start position (dragged from)
oldY	The y coordinate of the drag start position (dragged from)
newX	The x coordinate of the new position (dragged to)
newY	The y coordinate of the new position (dragged to)

7.65.2.2 \sim DragEvent()

```
~DragEvent ( ) [inline], [virtual]
```

Destructor.

7.65.3 Member Function Documentation

7.65.3.1 getDeltaX()

```
int16_t getDeltaX ( ) const [inline]
```

Gets the distance in x coordinates (how long was the drag).

Returns

The distance of this drag event.

7.65.3.2 getDeltaY()

```
int16_t getDeltaY ( ) const [inline]
```

Gets the distance in y coordinates (how long was the drag).

Returns

The distance of this drag event.

7.65.3.3 getEventType()

```
Event::EventType getEventType ( ) [inline], [virtual]
```

Gets event type.

Returns

The type of this event.

Implements Event.

```
7.65.3.4 getNewX()
int16_t getNewX ( ) const [inline]
Gets the new x coordinate (dragged to).
Returns
     The new x coordinate (dragged to).
7.65.3.5 getNewY()
int16_t getNewY ( ) const [inline]
Gets the new x coordinate (dragged to).
Returns
     The new y coordinate (dragged to).
7.65.3.6 getOldX()
int16_t getOldX ( ) const [inline]
Gets the x coordinate where the drag operation was started (dragged from).
Returns
     The x coordinate where the drag operation was started (dragged from).
7.65.3.7 getOldY()
int16_t getOldY ( ) const [inline]
Gets the y coordinate where the drag operation was started (dragged from).
Returns
     The y coordinate where the drag operation was started (dragged from).
7.65.3.8 getType()
DragEventType getType ( ) const [inline]
Gets the type of this drag event.
```

Returns

The type of this drag event.

7.66 Draggable < T > Class Template Reference

Mix-in class that extends a class to become draggable.

```
#include <touchgfx/mixins/Draggable.hpp>
```

Public Member Functions

• Draggable ()

Default constructor.

virtual ~Draggable ()

Destructor.

virtual void handleDragEvent (const DragEvent &evt)

Called when dragging the draggable object.

7.66.1 Detailed Description

```
template < class T > class touchgfx::Draggable < T >
```

Mix-in class that extends a class to become draggable.

Template Parameters

T | specifies the type to extend with the draggable behavior.

7.66.2 Constructor & Destructor Documentation

```
7.66.2.1 Draggable()
```

```
Draggable ( ) [inline]
```

Default constructor.

7.66.2.2 \sim Draggable()

```
~Draggable ( ) [inline], [virtual]
```

Destructor.

7.66.3 Member Function Documentation

7.66.3.1 handleDragEvent()

Called when dragging the draggable object. The object is moved according to the drag event.

Parameters

evt The drag event.

Reimplemented in Snapper< T >.

7.67 Drawable Class Reference

The Drawable class is an abstract definition of something that can be drawn.

#include <touchgfx/Drawable.hpp>

Public Types

enum DrawableType {

TYPE_DRAWABLE, TYPE_WIDGET, TYPE_ABSTRACTBUTTON, TYPE_ANIMATEDIMAGE, TYPE_B↔ OX, TYPE_BUTTON, TYPE_BUTTONWITHICON,

TYPE_BUTTONWITHLABEL, TYPE_IMAGE, TYPE_TILEDIMAGE, TYPE_KEYBOARD, TYPE_SCALA⇔ BLEIMAGE, TYPE_SNAPSHOTWIDGET, TYPE_TEXTAREA,

TYPE_TEXTAREAWITHONEWILDCARD, TYPE_TEXTAREAWITHTWOWILDCARDS, TYPE_TOGGLE
BUTTON, TYPE_TOUCHAREA, TYPE_CONTAINER, TYPE_LISTLAYOUT, TYPE_SCROLLABLECON
TAINER.

 $\label{type_zoomanimationimage} \textbf{TYPE_RADIOBUTTON}, \ \ \textbf{TYPE_TEXTUREMAPPER}, \ \ \textbf{TYPE_SLIDER}, \\ \textbf{TYPE_CUSTOMTYPESBEGIN}, \ \textbf{TYPE_CLICKABLECONTAINER} \ \}$

Enum defining Drawable types.

Public Member Functions

• Drawable ()

Default constructor.

virtual ∼Drawable ()

Destructor.

virtual void draw (const Rect &invalidatedArea) const =0

Pure virtual function for drawing this drawable.

• virtual Rect getSolidRect () const =0

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent).

· virtual void invalidateRect (Rect &invalidatedArea) const

Request that a subregion of this drawable is redrawn.

· virtual void invalidate () const

Tell the framework that this entire drawable needs to be redrawn.

Drawable * getNextSibling ()

Gets the next sibling.

virtual Rect getSolidRectAbsolute ()

Helper function for obtaining the largest solid rect.

• virtual void getLastChild (int16_t x, int16_t y, Drawable **last)=0

Function for obtaining the the last child of this drawable that intersects with the specified point.

· virtual void getVisibleRect (Rect &rect) const

JSMOC Function for obtaining the visible part of this drawable.

• const Rect & getRect () const

Gets the rectangle this Drawable covers.

Rect getAbsoluteRect () const

Helper function for obtaining the rectangle this Drawable covers.

virtual void translateRectToAbsolute (Rect &r) const

Helper function for converting a specified subregion of this Drawable to absolute coordinates.

virtual void setPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the size and position of this Drawable.

int16 t getX () const

Gets the x coordinate of this drawable.

int16_t getY () const

Gets the y coordinate of this drawable.

• int16_t getWidth () const

Gets the width of this drawable.

int16_t getHeight () const

Gets the height of this drawable.

virtual void setX (int16_t x)

Sets the x coordinate of this drawable.

virtual void setY (int16 t y)

Sets the y coordinate of this drawable.

virtual void setXY (int16_t x, int16_t y)

Sets the x and y coordinates of this drawable.

virtual void setWidth (int16 t width)

Sets the width of this drawable.

virtual void setHeight (int16_t height)

Sets the height of this drawable.

· virtual void childGeometryChanged ()

This function can be called on parent nodes to signal that the size of one or more of its children has changed.

virtual void handleClickEvent (const ClickEvent &evt)

Defines the event handler interface for ClickEvents.

virtual void handleGestureEvent (const GestureEvent &evt)

Defines the event handler interface for GestureEvents.

virtual void handleDragEvent (const DragEvent &evt)

Defines the event handler interface for DragEvents.

virtual void handleTickEvent ()

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

• void setVisible (bool vis)

Controls whether this Drawable should be visible.

void setTouchable (bool touch)

Controls whether this Drawable receives touch events or not.

• bool isVisible () const

Gets whether this Drawable is visible.

bool isTouchable () const

Gets whether this Drawable receives touch events or not.

Drawable * getParent ()

Returns the parent node.

virtual void moveRelative (int16_t x, int16_t y)

Moves the drawable.

virtual void moveTo (int16_t x, int16_t y)

Moves the drawable.

virtual uint16_t getType () const

For GUI testing only.

void drawToDynamicBitmap (BitmapId id)

Render the Drawable object into a dynamic bitmap.

Protected Member Functions

void resetDrawChainCache ()

For TouchGFX internal use only.

• Rect & getCachedVisibleRect ()

For TouchGFX internal use only.

int16_t getCachedAbsX ()

For TouchGFX internal use only.

int16_t getCachedAbsY ()

For TouchGFX internal use only.

virtual void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement)

For TouchGFX internal use only.

Protected Attributes

· Rect rect

The coordinates of this drawable, relative to its parent.

· Rect cachedVisibleRect

Cached representation of currently visible area. For TouchGFX internal use.

Drawable * parent

Pointer to this drawable's parent.

Drawable * nextSibling

Pointer to the next drawable. Maintained by containers.

Drawable * nextDrawChainElement

Next in draw chain. For TouchGFX internal use.

int16 t cachedAbsX

Cached value of absolute X-coord. For TouchGFX internal use.

int16 t cachedAbsY

Cached value of absolute Y-coord. For TouchGFX internal use.

bool touchable

True if this drawable should receive touch events.

· bool visible

True if this drawable should be drawn.

Static Protected Attributes

• static const int16_t UNCACHED_INDICATOR = -1

Constant representing uncached value. For TouchGFX internal use.

7.67.1 Detailed Description

The Drawable class is an abstract definition of something that can be drawn. In the composite design pattern, the Drawable is the component interface. Drawables can be added to a screen as a tree structure through the leaf node class Widget and the Container class. A Drawable contains a pointer to its next sibling and a pointer to its parent node. These are maintained by the Container to which the Drawable is added.

The Drawable interface contains two pure virtual functions which must be implemented by widgets, namely draw() and getSolidRect(). In addition it contains general functionality for receiving events and navigating the tree structure.

The coordinates of a Drawable are always relative to its parent node.

See also

Widget Container

7.67.2 Member Enumeration Documentation

7.67.2.1 DrawableType

```
enum enum DrawableType
```

Enum defining Drawable types. To be used by automated GUI testing to determine class type of a Drawable object.

7.67.3 Constructor & Destructor Documentation

```
7.67.3.1 Drawable()
Drawable ( ) [inline]
Default constructor.

7.67.3.2 ~Drawable()
~Drawable ( ) [inline], [virtual]
```

7.67.4 Member Function Documentation

7.67.4.1 childGeometryChanged()

Destructor.

```
void childGeometryChanged ( ) [inline], [virtual]
```

This function can be called on parent nodes to signal that the size of one or more of its children has changed. Currently only used in ScrollableContainer to redraw scrollbars when the size of the scrolling contents changes.

Reimplemented in ScrollableContainer.

Pure virtual function for drawing this drawable. It is a requirement that the draw implementation does not draw outside the region specified by invalidatedArea.

Parameters

invalidatedArea	The subregion of this drawable that needs to be redrawn, expressed in coordinates relative
	to its parent (e.g. for a complete redraw, invalidatedArea will be (0, 0, width, height).

Implemented in Keyboard, TextArea, TextAreaWithTwoWildcards, BoxWithBorder, ButtonWithLabel, ScalableImage, Container, TiledImage, ButtonWithIcon, Box, CanvasWidget, TextureMapper, TextAreaWithOneWildcard, Image,

RadioButton, SnapshotWidget, CoverTransition < templateDirection >::FullSolidRect, Button, PixelDataWidget, and TouchArea.

7.67.4.3 drawToDynamicBitmap()

Render the Drawable object into a dynamic bitmap.

Parameters

id The target dynamic bitmap to use for rendering.

7.67.4.4 getAbsoluteRect()

```
Rect getAbsoluteRect ( ) const
```

Helper function for obtaining the rectangle this Drawable covers, expressed in absolute coordinates.

Returns

The rectangle this Drawable covers expressed in absolute coordinates.

7.67.4.5 getCachedAbsX()

```
int16_t getCachedAbsX ( ) [inline], [protected]
```

Obtain cached version of absolute X-coord.

Note

For TouchGFX internal use only.

Returns

The absolute x coordinate for this drawable. Only calculated once.

7.67.4.6 getCachedAbsY()

```
int16_t getCachedAbsY ( ) [inline], [protected]
```

Obtain cached version of absolute Y-coord.

Note

For TouchGFX internal use only.

Returns

The absolute y coordinate for this drawable. Only calculated once.

7.67.4.7 getCachedVisibleRect()

```
Rect & getCachedVisibleRect ( ) [inline], [protected]
```

Obtain cached version of visible rect.

Note

For TouchGFX internal use only.

Returns

The Visible rect for this drawable. Only calculated once.

7.67.4.8 getHeight()

```
int16_t getHeight ( ) const [inline]
```

Gets the height of this drawable.

Returns

The height of this drawable.

7.67.4.9 getLastChild()

```
void getLastChild (
    int16_t x,
    int16_t y,
    Drawable ** last ) [pure virtual]
```

Function for obtaining the the last child of this drawable that intersects with the specified point. Used in input event handling for obtaining the appropriate drawable that should receive the event. Note that input events must be delegated to the last drawable of the tree (meaning highest z-order / front-most drawable).

Parameters

	Х	The point of intersection expressed in coordinates relative to the parent.	
	У	The point of intersection expressed in coordinates relative to the parent.	
out	last	Result will be placed here.	

Implemented in ScrollableContainer, Container, and Widget.

7.67.4.10 getNextSibling()

```
Drawable * getNextSibling ( ) [inline]
```

Returns the next sibling node. This will be the next Drawable that has been added to the same Container as this Drawable.

Returns

The next sibling. If this is the last sibling, the return value is 0.

7.67.4.11 getParent()

```
Drawable * getParent ( ) [inline]
```

Returns the parent node. For the root container, the return value is 0.

Returns

The parent node. For the root container, the return value is 0.

7.67.4.12 getRect()

```
const Rect & getRect ( ) const [inline]
```

Gets the rectangle this Drawable covers, in coordinates relative to its parent.

Returns

The rectangle this Drawable covers expressed in coordinates relative to its parent.

7.67.4.13 getSolidRect()

```
Rect getSolidRect ( ) const [pure virtual]
```

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent). Used by JSMOC to prune the draw graph.

Note

The rectangle returned must be relative to (0, 0), meaning that to indicate a completely solid widget, Rect(0, 0, getWidth(), getHeight()) must be returned.

Returns

The solid rect.

Implemented in ScalableImage, CanvasWidget, Container, TiledImage, ButtonWithLabel, TextureMapper, Image, RadioButton, TouchArea, Button, SnapshotWidget, Box, BoxWithBorder, PixeIDataWidget, TextArea, and Cover

Transition< templateDirection >::FullSolidRect.

7.67.4.14 getSolidRectAbsolute()

```
Rect getSolidRectAbsolute ( ) [virtual]
```

Helper function for obtaining the largest solid rect (as implemented by getSolidRect()) expressed in absolute coordinates. Will recursively traverse to the root of the tree.

Returns

Largest solid rect (as implemented by getSolidRect()) expressed in absolute coordinates.

7.67.4.15 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Call this virtual function to determine the class type of this Drawable object. Can be used in automated GUI testing. Otherwise this function is unused.

Note

If creating custom drawables that need to be manipulated by a test framework, override this function and return a custom value higher than or equal to TYPE_CUSTOMTYPESBEGIN.

Returns

An integer describing the class type of this object, corresponding to the DrawableType enum for built-in Drawables.

Reimplemented in TextureMapper, TextArea, ZoomAnimationImage, ScrollableContainer, Slider, TextAreaWith

TwoWildcards, Keyboard, RadioButton, AnimatedImage, Container, ScalableImage, BoxWithBorder, ButtonWith

Label, TiledImage, TextAreaWithOneWildcard, Box, ButtonWithIcon, ListLayout, Image, SnapshotWidget, Button,

TouchArea, ToggleButton, AbstractButton, and Widget.

7.67.4.16 getVisibleRect()

JSMOC Function for obtaining the visible part of this drawable. If the parent node has a smaller area than this Drawable, the parent will act as a viewport, cutting off the parts of this Drawable that are outside the region. Traverses the tree and yields a result expressed in absolute coordinates.

Parameters

out	rect	The subregion of the drawable on which to perform the operation.
-----	------	--

7.67.4.17 getWidth()

```
int16_t getWidth ( ) const [inline]
```

Gets the width of this drawable.

Returns

The width of this drawable.

```
7.67.4.18 getX()
```

```
int16_t getX ( ) const [inline]
```

Gets the x coordinate of this drawable.

Returns

The x value, relative to the parent.

7.67.4.19 getY()

```
int16_t getY ( ) const [inline]
```

Gets the y coordinate of this drawable.

Returns

The y value, relative to the parent.

7.67.4.20 handleClickEvent()

Defines the event handler interface for ClickEvents. The default implementation ignores the event. The event is only received if the drawable is touchable.

Parameters

```
evt The ClickEvent received from the HAL.
```

Reimplemented in Slider, Keyboard, ScrollableContainer, ScrollUist, ScrollWheelBase, RepeatButton, Repeat ButtonTrigger, ToggleButton, ToggleButtonTrigger, RadioButton, TouchArea, AbstractButton, ClickButtonTrigger, TouchButtonTrigger, and SwipeContainer.

7.67.4.21 handleDragEvent()

Defines the event handler interface for DragEvents. The event is only received if the drawable is touchable.

Parameters

evt The DragEvent received from the HAL.

Reimplemented in ScrollBase, Slider, Keyboard, ScrollableContainer, ScrollWheelBase, TouchArea, and Swipe Container.

7.67.4.22 handleGestureEvent()

```
void handleGestureEvent (
                      const GestureEvent & evt ) [inline], [virtual]
```

Defines the event handler interface for GestureEvents. The default implementation ignores the event. The event is only received if the drawable is touchable.

Parameters

```
evt The GestureEvent received from the HAL.
```

Reimplemented in ScrollBase, ScrollableContainer, ScrollWheelBase, and SwipeContainer.

7.67.4.23 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

See also

Application::registerTimerWidget

Reimplemented in ScrollBase, SlideMenu, MoveAnimator< touchgfx::Container >, ScrollableContainer, Zoom AnimationImage, AnimationTextureMapper, RepeatButtonTrigger, RepeatButton, AnimatedImage, and Swipe Container.

7.67.4.24 invalidate()

```
void invalidate ( ) const [virtual]
```

Tell the framework that this entire drawable needs to be redrawn.

See also

invalidateRect

Reimplemented in CanvasWidget.

7.67.4.25 invalidateRect()

Request that a subregion of this drawable is redrawn. Will recursively traverse the tree towards the root, and once reached, issue a draw operation. When this function returns, the specified invalidated area has been redrawn for all appropriate Drawables covering the region.

Parameters

in	invalidatedArea	The area of this drawable to redraw expressed in coordinates relative to its parent (e.g.	1
		to request a complete redraw, invalidatedArea will be (0, 0, width, height).	

Reimplemented in CacheableContainer.

```
7.67.4.26 isTouchable()
bool isTouchable ( ) const [inline]
```

Gets whether this Drawable receives touch events or not.

Returns

True if touch events are received.

See also

setTouchable

```
7.67.4.27 isVisible()
```

```
bool isVisible ( ) const [inline]
```

Gets whether this Drawable is visible.

Returns

true if the Drawable is visible. True if visible.

See also

setVisible

7.67.4.28 moveRelative()

```
void moveRelative ( \label{eq:continuity} \verb"int16_t x", \\ \verb"int16_t y") \quad [virtual]
```

Moves the drawable.

Note

Will redraw the appropriate areas of the screen.

Parameters

X	The relative position to move to.
у	The relative position to move to.

7.67.4.29 moveTo()

Moves the drawable.

Note

Will redraw the appropriate areas of the screen.

Parameters

X	The absolute position to move to.
У	The absolute position to move to.

7.67.4.30 resetDrawChainCache()

```
void resetDrawChainCache ( ) [inline], [protected]
```

Reset cached coordinate data.

Note

For TouchGFX internal use only.

7.67.4.31 setHeight()

Sets the height of this drawable.

Note

Changing this does not automatically yield a redraw.

Parameters

```
height The new height.
```

Reimplemented in ZoomAnimationImage, DrawableList, DigitalClock, ScrollWheelWithSelectionStyle, and Scroll← Base.

7.67.4.32 setPosition()

```
void setPosition ( int16_t x, int16_t y,
```

```
int16_t width,
int16_t height ) [inline], [virtual]
```

Sets the size and position of this Drawable, relative to its parent.

Note

Changing this does not automatically yield a redraw.

Parameters

X	The x coordinate of this Drawable.
У	The y coordinate of this Drawable.
width	The width of this Drawable.
height	The height of this Drawable.

Reimplemented in ZoomAnimationImage.

7.67.4.33 setTouchable()

```
void setTouchable (
          bool touch ) [inline]
```

Controls whether this Drawable receives touch events or not.

Parameters

7.67.4.34 setupDrawChain()

Configure linked list for draw chain.

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

Reimplemented in Keyboard, Container, CacheableContainer::CachedImage, and CacheableContainer.

7.67.4.35 setVisible()

```
void setVisible ( bool\ vis\ )\quad [inline]
```

Controls whether this <u>Drawable</u> should be visible. Only visible Drawables will have their draw function called. Additionally, invisible drawables will not receive input events.

Note

Changing this does not automatically yield a redraw.

Parameters

vis

true if this Drawable should be visible. By default, drawables are visible unless this function has been called with false as argument.

7.67.4.36 setWidth()

```
void setWidth (
          int16_t width ) [inline], [virtual]
```

Sets the width of this drawable.

Note

Changing this does not automatically yield a redraw.

Parameters

```
width The new width.
```

Reimplemented in ZoomAnimationImage, DrawableList, DigitalClock, ScrollBase, and ScrollWheelWithSelection← Style.

7.67.4.37 setX()

Sets the x coordinate of this drawable.

Note

Changing this does not automatically yield a redraw.

Parameters

x The new x value, relative to the parent. A negative value is allowed.

7.67.4.38 setXY()

Sets the x and y coordinates of this drawable.

Note

Changing this does not automatically yield a redraw.

Parameters

X	The new x value, relative to the parent. A negative value is allowed.
У	The new y value, relative to the parent. A negative value is allowed.

7.67.4.39 setY()

Sets the y coordinate of this drawable.

Note

Changing this does not automatically yield a redraw.

Parameters

y The new y value, relative to the parent. A negative value is allowed.

7.67.4.40 translateRectToAbsolute()

Helper function for converting a specified subregion of this Drawable to absolute coordinates.

Parameters

in, out r	The Rect to translate.
-----------	------------------------

7.68 DrawableList Class Reference

A container able to display many items using only a few drawables.

#include <touchgfx/containers/DrawableList.hpp>

Public Member Functions

· DrawableList ()

Default constructor.

virtual ~DrawableList ()

Destructor.

virtual void setWidth (int16_t width)

Sets the width of the DrawableList.

virtual void setHeight (int16_t height)

Sets the height of the DrawableList. If the list is horizontal, the height is also propagated to all drawables in the list.

virtual void setHorizontal (bool horizontal)

Sets a horizontal layout.

· virtual bool getHorizontal () const

Gets the orientation of the drawables.

virtual void setCircular (bool circular)

Sets whether the list is circular or not.

· virtual bool getCircular () const

Gets the circular setting.

void setDrawableSize (int16 t drawableSize, int16 t drawableMargin)

Sets drawable size.

• virtual int16_t getItemSize () const

Gets size of each item.

• virtual int16_t getDrawableSize () const

Gets drawable size.

virtual int16_t getDrawableMargin () const

Gets drawable margin.

virtual void setDrawables (DrawableListItemsInterface &drawableListItems, int16_t drawableItemIndexOffset,
 GenericCallback
 DrawableListItemsInterface *, int16_t, int16_t > &updateDrawableCallback)

Sets the drawables parameters.

• int16_t getNumberOfDrawables () const

Gets number of drawables.

void setNumberOfItems (int16_t numberOfItems)

Sets number of items in the list.

• int16_t getNumberOfItems () const

Gets number of items in the DrawableList.

int16_t getRequiredNumberOfDrawables () const

Gets required number of drawables.

void setOffset (int32_t ofs)

Sets virtual coordinate.

int32_t getOffset () const

Gets offset.

int16_t getItemIndex (int16_t drawableIndex) const

Gets item stored in a given Drawable.

• int16_t getDrawableIndices (int16_t itemIndex, int16_t *drawableIndexArray, int16_t arraySize) const Gets drawable indices.

• int16_t getDrawableIndex (int16_t itemIndex, int16_t prevDrawableIndex=-1) const

Gets the drawable index of an item.

• void refreshDrawables ()

Refresh drawables.

void itemChanged (int16_t itemIndex)

Item changed.

Additional Inherited Members

7.68.1 Detailed Description

A container able to display many items using only a few drawables. This is done by only having drawables for visible items, and populating these drawables with new content when the drawable becomes visible.

This means that all drawables must have an identical structure in some way, for example an Image or a Container with a button and a text.

7.68.2 Constructor & Destructor Documentation

```
7.68.2.1 DrawableList()
DrawableList ( )
Default constructor.
7.68.2.2 ~DrawableList()
~DrawableList ( ) [inline], [virtual]
```

7.68.3 Member Function Documentation

```
7.68.3.1 getCircular()
bool getCircular ( ) const [virtual]
```

Gets the circular setting, previously set using setCircular().

Returns

Destructor.

True if the list is circular (infinite), false if the list is not circular.

See also

setCircular

7.68.3.2 getDrawableIndex()

Gets the drawable index of an item. If the number of items is smaller than the number of drawables and the DrawableList is circular, the same item can be in more than one drawable. In that case, calling this function again with the previously returned index as second parameter, the index of the next drawable containing the item will be returned.

Parameters

itemIndex	Index of the item.
prevDrawableIndex	(Optional) Index of the previous drawable. If given, search starts after this index.

Returns

The first drawable index with the given item. Returns -1 if the item is not in a drawable.

See also

getDrawableIndices

7.68.3.3 getDrawableIndices()

Gets drawable indices. Useful when the number of items is smaller than the number of drawables as the same item might be in more than one drawable on the screen (if the <u>DrawableList</u> is circular). The passed array will be filled with the drawable indices and the number of indices found is returned.

Parameters

	itemIndex	Zero-based index of the item.
out	drawableIndexArray	Array where the drawable indices are stored.
	arraySize	Size of drawable array.

Returns

The number of drawable indices found.

See also

```
getFirstDrawableIndex
setCircular
getDrawbleIndex
```

7.68.3.4 getDrawableMargin()

```
int16_t getDrawableMargin ( ) const [virtual]
```

Gets drawable margin as set by setDrawables.

Returns

The drawable margin.

7.68.3.5 getDrawableSize()

```
int16_t getDrawableSize ( ) const [virtual]
```

Gets drawable size as set by setDrawables.

Returns

The drawable size.

See also

setDrawables

7.68.3.6 getHorizontal()

```
bool getHorizontal ( ) const [virtual]
```

Gets the orientation of the drawables, previously set using setHorizontal().

Returns

True if it horizontal, false if it is vertical.

See also

setHorizontal

7.68.3.7 getItemIndex()

Gets item stored in a given Drawable.

Parameters

drawahlaladay	Zara based index of the drawable
urawabieiriuex	Zero-based index of the drawable.

Returns

The item index.

7.68.3.8 getItemSize()

```
int16_t getItemSize ( ) const [virtual]
```

Gets size of each item. This equals the drawable size plus the drawable margin as set in setDrawables(). Equals getDrawableSize() + getDrawableMargin().

```
Returns
     The item size.
Note
     Not the same as getDrawableSize().
See also
     setDrawables
     setDrawableSize
     getDrawableMargin
7.68.3.9 getNumberOfDrawables()
int16_t getNumberOfDrawables ( ) const
Gets number of drawables, as set using setDrawables().
Returns
     The number of drawables.
See also
     setDrawables
7.68.3.10 getNumberOfItems()
int16_t getNumberOfItems ( ) const
Gets number of items in the DrawableList, as previously set using setNumberOfItems().
Returns
     The number of items.
See also
     setNumberOfItems
7.68.3.11 getOffset()
int32_t getOffset ( ) const
Gets offset, as previously set using setOffset().
Returns
     The virtual offset.
See also
```

setOffset

7.68.3.12 getRequiredNumberOfDrawables()

```
int16_t getRequiredNumberOfDrawables ( ) const
```

Gets required number of drawables. After setting up the DrawableList it is possible to request how many drawables are needed to ensure that the list can always be drawn properly. If the DrawableList has been setup with fewer Drawables than the required number of drawables, part of the lower part of the DrawableList will look wrong.

The number of required drawables depend on the size of the widget and the size of the drawables and the margin around drawables. If there are fewer drawables than required, the widget will not display correctly. If there are more drawables than required, some will be left unused.

Returns

The required number of drawables.

See also

setDrawables

7.68.3.13 itemChanged()

Item changed and drawables containing this item must be updated. This function can be called when an item has changed and needs to be updated on screen. If the given item is displayed on screen, possible more than once for cyclic lists, each drawable is request to refresh its content to reflect the new value.

Parameters

```
itemIndex Zero-based index of the item.
```

7.68.3.14 refreshDrawables()

```
void refreshDrawables ( )
```

Refresh drawables. Useful to call if the number or items, their size or other properties have changed.

7.68.3.15 setCircular()

Sets whether the list is circular (infinite) or not. A circular list is a list where the first drawable re-appears after the last item in the list.

Parameters

See also

getCircular

7.68.3.16 setDrawables()

Sets the drawables parameters. These parameters are

- · The access class to the array of drawables
- · The offset in the drawableListItems array to start using drawable and
- Callback to update the contents of a drawable.

Parameters

in,out	drawableListItems	Number of drawables allocated.
in	drawableItemIndexOffset	A callback to get access to a drawable.
in	updateDrawableCallback	A callback to update the contents of a drawable.

See also

getRequiredNumberOfDrawables

7.68.3.17 setDrawableSize()

Sets drawable size. The total size of each drawable is the drawableSize + 2*drawableMargin as margin will be added before and after each drawable.

Parameters

drawableSize	The size of the drawable.
drawableMargin	The margin around drawables (before as well as after the drawable).

7.68.3.18 setHeight()

Parameters

<i>height</i> ∣ The height.	height	The height.
-----------------------------	--------	-------------

Reimplemented from Drawable.

7.68.3.19 setHorizontal()

Sets a horizontal layout. If horizontal is set true, all drawables are arranged side by side. If horizontal is set false, the drawables are arranged above and below each other (vertically).

Parameters

horizontal	True to align drawables horizontal, false to align drawables vertically.
------------	--

Note

Default value is false, i.e. vertical layout.

See also

getHorizontal

7.68.3.20 setNumberOfItems()

Sets number of items in the list. This forces all drawables to be updated to ensure that the content is correct.

Parameters

numberOfItems	Number of items.
---------------	------------------

Note

The DrawableList is refreshed to reflect the change.

7.68.3.21 setOffset()

```
void setOffset (
          int32_t ofs )
```

Sets virtual coordinate. Does not move to the given coordinate, but places the drawables and fill correct content into the drawables to give the impression that everything has been scrolled to the given coordinate.

Setting a value of 0 means that item 0 is at the start of the DrawableList. Setting a value of "-getItemSize()" places item 0 outside the start of the DrawableList and item 1 at the start of it.

Items that are completely outside of view, will be updated with new content using the provided callback from set← Drawables(). Care is taken to not fill drawables more than strictly required.

Parameters

```
ofs The virtual coordinate.
```

See also

getOffset setDrawables

7.68.3.22 setWidth()

```
void setWidth (
                int16_t width ) [virtual]
```

Sets the width of the DrawableList. If the list is vertical, the width is also propagated to all drawables in the list.

Parameters

width	The width.
wiatn	i ne wiath.

Reimplemented from Drawable.

7.69 DrawableListItems < TYPE, SIZE > Class Template Reference

An array of drawables used by DrawableList.

```
#include <touchgfx/containers/DrawableList.hpp>
```

Public Member Functions

• DrawableListItems ()

Default constructor.

virtual ∼DrawableListItems ()

Destructor.

virtual Drawable * getDrawable (int16_t index)

Gets the address of an element.

• TYPE & operator[] (int index)

Array indexer operator.

• virtual int16_t getNumberOfDrawables ()

Gets number of drawables.

Public Attributes

• TYPE element [SIZE]

The array of drawables.

7.69.1 Detailed Description

```
\label{template} \begin{tabular}{ll} template < class TYPE, int SIZE > \\ class touchgfx::DrawableListItems < TYPE, SIZE > \\ \end{tabular}
```

An array of drawables used by DrawableList. This class is primarily used to ease the setup of a callback function to get access to a specific drawable in the array.

Example usage:

```
static const int NUMBER_OF_DRAWABLES = 5;
DrawableListItems<TextAreaWithOneWildcardContainer, NUMBER_OF_DRAWABLES> menuItems;
```

Template Parameters

TYPE	Type of the drawables. Can be a simple drawable, such as Image or a more complex container.
SIZE	Size of the array. This is the number of drawables to allocate and should be all visible drawables on
	the screen at any given time.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 DrawableListItems()

```
DrawableListItems ( ) [inline]
```

Default constructor.

7.69.2.2 \sim DrawableListItems()

```
~DrawableListItems ( ) [inline], [virtual]
```

Destructor.

7.69.3 Member Function Documentation

7.69.3.1 getDrawable()

```
Drawable * getDrawable (
                int16_t index ) [inline], [virtual]
```

Gets the address of an element.

Parameters

index	Zero-based index of the drawable.
-------	-----------------------------------

Returns

The drawable for the given index.

Implements DrawableListItemsInterface.

7.69.3.2 getNumberOfDrawables()

```
virtual int16_t getNumberOfDrawables ( ) [inline], [virtual]
```

Gets number of drawables.

Returns

The number of drawables.

Implements DrawableListItemsInterface.

7.69.3.3 operator[]()

```
TYPE & operator[] (
          int index ) [inline]
```

Array indexer operator.

Parameters

index Zero-based index of elements to access.

Returns

The indexed value.

7.70 DrawableListItemsInterface Class Reference

A drawable list items interface.

```
#include <touchgfx/containers/DrawableList.hpp>
```

Public Member Functions

- virtual \sim DrawableListItemsInterface ()

Destructor.

• virtual Drawable * getDrawable (int16_t index)=0

Gets a drawable.

• virtual int16_t getNumberOfDrawables ()=0

Gets number of drawables.

7.70.1 Detailed Description

A drawable list items interface. Used to pass the allocated array of drawable elements to setDrawables function in either ScrollUist, ScrollWheel or ScrollWheelWithSelectionStyle. Provides easy access to each element in the array as well as the size of the array.

See also

ScrollList::setDrawables ScrollWheel::setDrawables

ScrollWheelWithSelectionStyle::setDrawables

7.70.2 Constructor & Destructor Documentation

```
7.70.2.1 ~DrawableListItemsInterface()
```

```
\simDrawableListItemsInterface ( ) [inline], [virtual]
```

Destuctor.

7.70.3 Member Function Documentation

7.70.3.1 getDrawable()

```
Drawable * getDrawable (
                      int16_t index ) [pure virtual]
```

Gets a drawable at a given index.

Parameters

index Zero-based index of the drawable.

Returns

Null if it fails, else the drawable.

Implemented in DrawableListItems < TYPE, SIZE >.

7.70.3.2 getNumberOfDrawables()

```
int16_t getNumberOfDrawables ( ) [pure virtual]
```

Gets number of drawables.

Returns

The number of drawables.

Implemented in DrawableListItems < TYPE, SIZE >.

7.71 DrawingSurface Struct Reference

The destination of a draw operation. Contains a pointer to where to draw and the stride of the drawing surface.

```
#include <touchgfx/hal/Types.hpp>
```

Public Attributes

uint16 t * address

The bits.

· int32 t stride

The stride.

7.72 Bitmap::DynamicBitmapData Struct Reference

Data of a dynamic bitmap.

```
#include <touchgfx/Bitmap.hpp>
```

Public Attributes

· Rect solid

The solidRect of this bitmap.

• uint16_t width

The width of the bitmap.

uint16_t height

The height of the bitmap.

• uint8 t format: 5

Determine the format of the data.

• uint8_t inuse: 1

Zero if not in use.

• uint8_t extra: 2

Extra data field, dependent on format.

7.72.1 Detailed Description

Data of a dynamic bitmap.

7.73 Easing Equations Class Reference

Defines the "Penner easing functions", which are a de facto standard computing aesthetically pleasing motion animations.

```
#include <touchgfx/EasingEquations.hpp>
```

Static Public Member Functions

• static int16_t backEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)

Back easing in: Overshooting cubic easing: $(s+1)*t^3 - s*t^2$.

static int16 t backEaseOut (uint16 t t, int16 t b, int16 t c, uint16 t d)

Back easing out: Overshooting cubic easing: $(s+1)*t^{\wedge}3 - s*t^{\wedge}2$.

• static int16_t backEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

Back easing in/out: Overshooting cubic easing: $(s+1)*t^3 - s*t^2$.

• static int16 t bounceEaseIn (uint16 t t, int16 t b, int16 t c, uint16 t d)

Bounce easing in - exponentially decaying parabolic bounce.

static int16_t bounceEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

Bounce easing out - exponentially decaying parabolic bounce.

• static int16_t bounceEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

Bounce easing in/out - exponentially decaying parabolic bounce.

- static int16_t circEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Circular easing in: sqrt(1-t^2)
- static int16_t circEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Circular easing out: sqrt(1-t^2)
- static int16_t circEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Circular easing in/out: sqrt(1-t\^2)
- static int16_t cubicEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Cubic easing in: t[^]3.
- static int16_t cubicEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Cubic easing out: t^3.
- static int16_t cubicEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Cubic easing in/out: t^3.
- static int16_t elasticEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Elastic easing in exponentially decaying sine wave.
- static int16_t elasticEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Elastic easing out exponentially decaying sine wave.
- static int16_t elasticEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Elastic easing in/out exponentially decaying sine wave.
- static int16_t expoEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Exponential easing in: 2[^]t.
- static int16_t expoEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Exponential easing out: 2[^]t.
- static int16_t expoEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Exponential easing in/out: 2[^]t.
- static int16_t linearEaseNone (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Simple linear tweening no easing.
- static int16_t linearEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Simple linear tweening no easing.
- static int16_t linearEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Simple linear tweening no easing.
- static int16_t linearEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)

 Simple linear tweening no easing.
- static int16_t quadEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quadratic easing in: t^2.
- static int16_t quadEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quadratic easing out: t^2.
- static int16_t quadEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quadratic easing in/out: t\(^2 2\).
- static int16_t quartEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quartic easing in: t[^]4.
- static int16_t quartEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quartic easing out: t^4.
- static int16_t quartEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quartic easing in/out: t^4.
- static int16_t quintEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quintic/strong easing in: t\(^5\).
- static int16_t quintEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
 Quintic/strong easing out: t^5.

```
    static int16_t quintEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
        Quintic/strong easing in/out: t^5.
    static int16_t sineEaseIn (uint16_t t, int16_t b, int16_t c, uint16_t d)
        Sinusoidal easing in: sin(t)
    static int16_t sineEaseOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
        Sinusoidal easing out: sin(t)
    static int16_t sineEaseInOut (uint16_t t, int16_t b, int16_t c, uint16_t d)
        Sinusoidal easing in/out: sin(t)
```

7.73.1 Detailed Description

Defines the "Penner easing functions", which are a de facto standard computing aesthetically pleasing motion animations. See http://easings.net/ for visual illustrations of the easing equations.

See also

```
http://easings.net/
```

7.73.2 Member Function Documentation

7.73.2.1 backEaseIn()

Back easing in: Overshooting cubic easing: $(s+1)*t^3 - s*t^2$. Backtracking slightly, then reversing direction and moving to target.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.2 backEaseInOut()

Back easing in/out: Overshooting cubic easing: $(s+1)*t^3 - s*t^2$. Backtracking slightly, then reversing direction and moving to target, then overshooting target, reversing, and finally coming back to target.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.3 backEaseOut()

Back easing out: Overshooting cubic easing: $(s+1)*t^3 - s*t^2$. Moving towards target, overshooting it slightly, then reversing and coming back to target.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.4 bounceEaseIn()

Bounce easing in - exponentially decaying parabolic bounce.

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.5 bounceEaseInOut()

Bounce easing in/out - exponentially decaying parabolic bounce.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.6 bounceEaseOut()

Bounce easing out - exponentially decaying parabolic bounce.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.7 circEaseIn()

```
int16_t b,
int16_t c,
uint16_t d ) [static]
```

Circular easing in: $sqrt(1-t^2)$. Accelerating from zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.8 circEaseInOut()

Circular easing in/out: $sqrt(1-t^2)$. Acceleration until halfway, then deceleration.

Parameters

-	
t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.9 circEaseOut()

Circular easing out: $sqrt(1-t^{\wedge}2)$. Decelerating to zero velocity.

t	Time. The current time or step.
b	Beginning. The beginning value.

Parameters

С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.10 cubicEaseIn()

Cubic easing in: t^{\(\)}3. Accelerating from zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.11 cubicEaseInOut()

Cubic easing in/out: t^3 . Acceleration until halfway, then deceleration.

t	Time. The current time or step.	
b	Beginning. The beginning value.	
С	Change. The change between the beginning value and the destination value.	
d	Duration. The total time or total number of steps.	

Returns

The current value as a function of the current time or step.

7.73.2.12 cubicEaseOut()

Cubic easing out: t^{\(\)}3. Decelerating to zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.13 elasticEaseIn()

Elastic easing in - exponentially decaying sine wave.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.14 elasticEaseInOut()

```
int16_t b,
int16_t c,
uint16_t d ) [static]
```

Elastic easing in/out - exponentially decaying sine wave.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.15 elasticEaseOut()

Elastic easing out - exponentially decaying sine wave.

Parameters

t	Time. The current time or step.	
b	Beginning. The beginning value.	
С	Change. The change between the beginning value and the destination value.	
d	Duration. The total time or total number of steps.	

Returns

The current value as a function of the current time or step.

7.73.2.16 expoEaseIn()

Exponential easing in: 2[^]t. Accelerating from zero velocity.

-		
t	Time. The current time or step.	
b	Beginning. The beginning value.	

Parameters

С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.17 expoEaseInOut()

Exponential easing in/out: 2^t. Accelerating until halfway, then decelerating.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.18 expoEaseOut()

Exponential easing out: 2^t. Deceleration to zero velocity.

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.19 linearEaseIn()

Simple linear tweening - no easing.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.20 linearEaseInOut()

Simple linear tweening - no easing.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.21 linearEaseNone()

```
int16_t b,
int16_t c,
uint16_t d ) [static]
```

Simple linear tweening - no easing.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.22 linearEaseOut()

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.23 quadEaseIn()

Quadratic easing in: t^2 . Accelerating from zero velocity.

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.24 quadEaseInOut()

Quadratic easing in/out: t^2 . Acceleration until halfway, then deceleration.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.25 quadEaseOut()

Quadratic easing out: $t^{\wedge}2$. Decelerating to zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.26 quartEaseIn()

```
static int16_t quartEaseIn ( \label{eq:continuous} \mbox{uint16\_t} \ t \mbox{,}
```

```
int16_t b,
int16_t c,
uint16_t d ) [static]
```

Quartic easing in: t⁴. Accelerating from zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.27 quartEaseInOut()

Quartic easing in/out: t^4 . Acceleration until halfway, then deceleration.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.28 quartEaseOut()

Quartic easing out: t^4 . Decelerating to zero velocity.

t	Time. The current time or step.
b	Beginning. The beginning value.

Parameters

С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.29 quintEaseIn()

Quintic/strong easing in: t⁵. Accelerating from zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.30 quintEaseInOut()

Quintic/strong easing in/out: t^5 . Acceleration until halfway, then deceleration.

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.31 quintEaseOut()

Quintic/strong easing out: t[^]5. Decelerating to zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.32 sineEaseIn()

Sinusoidal easing in: sin(t). Accelerating from zero velocity.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.33 sineEaseInOut()

```
int16_t b,
int16_t c,
uint16_t d ) [static]
```

Sinusoidal easing in/out: sin(t). Acceleration until halfway, then deceleration.

Parameters

t	Time. The current time or step.
b	Beginning. The beginning value.
С	Change. The change between the beginning value and the destination value.
d	Duration. The total time or total number of steps.

Returns

The current value as a function of the current time or step.

7.73.2.34 sineEaseOut()

Sinusoidal easing out: sin(t). Decelerating to zero velocity.

Parameters

-		
t	Time. The current time or step.	
b	Beginning. The beginning value.	
С	Change. The change between the beginning value and the destination value.	
d	Duration. The total time or total number of steps.	

Returns

The current value as a function of the current time or step.

7.74 Edge Struct Reference

An edge contains information about one edge, between two points, of a triangle, as well as information about how to interpolate values when moving in the vertical direction.

```
#include <touchgfx/TextureMapTypes.hpp>
```

Public Member Functions

- Edge (const Gradients &gradients, const Point3D *vertices, int top, int bottom)
 Constructor.
- int step ()

Perform a step along the edge.

int step (int steps)

Performs a number of steps along the edge.

Public Attributes

int32_t X

The X coordinate.

int32_t XStep

Amount to increment x.

• int32_t numerator

The numerator.

· int32_t denominator

The denominator.

• int32_t errorTerm

The error term.

int Y

The Y coordinate.

· int height

The height.

float oneOverZ

The one over z coordinate.

float oneOverZStep

The one over z coordinate step.

float oneOverZStepExtra

The one over z coordinate step extra.

float UOverZ

The over z coordinate.

float UOverZStep

The over z coordinate step.

• float UOverZStepExtra

The over z coordinate step extra.

float VOverZ

The over z coordinate.

float VOverZStep

The over z coordinate step.

float VOverZStepExtra

The over z coordinate step extra.

7.74.1 Constructor & Destructor Documentation

Construct the edge between two vertices and using the gradients for calculating the interpolation values.

Parameters

gradients	The gradients for the triangle.
vertices	The vertices for the triangle.
top	The index in the vertices array of the top vertex of this edge.
bottom	The index in the vertices array of the bottom vertex of this edge.

7.74.2 Member Function Documentation

```
7.74.2.1 step() [1/2]
int step ( ) [inline]
```

Perform a step along the edge.

Returns

the Height.

Performs a number of steps along the edge.

Parameters

steps	The number of steps the perform.
-------	----------------------------------

Returns

height.

7.75 Event Class Reference

Simple base class for events.

```
#include <touchgfx/Event.hpp>
```

Public Types

enum EventType { EVENT_CLICK, EVENT_DRAG, EVENT_GESTURE }
 The events types.

Public Member Functions

• virtual EventType getEventType ()=0

Gets event type.

virtual ∼Event ()

Destructor.

7.75.1 Detailed Description

Simple base class for events.

7.75.2 Member Enumeration Documentation

7.75.2.1 EventType

enum enum EventType

The events types.

Enumerator

EVENT_CLICK	A click.
EVENT_DRAG	A drag.
EVENT_GESTURE	A gesture.

7.75.3 Constructor & Destructor Documentation

7.75.4 Member Function Documentation

```
7.75.4.1 getEventType()
```

```
EventType getEventType ( ) [pure virtual]
```

Gets event type.

Destructor.

Returns

The type of this event.

Implemented in ClickEvent, DragEvent, and GestureEvent.

7.76 FadeAnimator < T > Class Template Reference

A FadeAnimator makes the template class T able to animate an alpha fade.

#include <touchgfx/mixins/FadeAnimator.hpp>

Public Member Functions

FadeAnimator ()

Default constructor.

virtual ∼FadeAnimator ()

Destructor.

void setFadeAnimationEndedAction (GenericCallback< const FadeAnimator< T > & > &callback)

Associates an action to be performed when the animation ends.

void clearFadeAnimationEndedAction ()

Clears the fade animation ended action previously set by setFadeAnimationEndedAction.

virtual void setFadeAnimationDelay (uint16_t delay)

Sets a delay on animations done by the FadeAnimator.

virtual uint16_t getFadeAnimationDelay () const

Gets the current animation delay.

virtual bool isRunning () const

Gets whether or not the fade animation is running.

· virtual bool isFadeAnimationRunning () const

Gets whether or not the fade animation is running.

Starts the fade animation.

void cancelFadeAnimation ()

Cancel fade animation.

Protected Member Functions

virtual void handleTickEvent ()

The tick handler that handles the actual animation steps.

void nextFadeAnimationStep ()

Execute next step in fade animation.

Protected Attributes

bool fadeAnimationRunning

Boolean that is true if the animation is running.

uint16 t fadeAnimationCounter

Counter that is equal to the current step in the animation.

uint16_t fadeAnimationDelay

A delay that is applied before animation start. Expressed in ticks.

uint16_t fadeAnimationDuration

The complete duration of the animation. Expressed in ticks.

int16 t fadeAnimationStartAlpha

The alpha value at the beginning of the animation.

int16_t fadeAnimationEndAlpha

The alpha value at the end of the animation.

EasingEquation fadeAnimationAlphaEquation

EasingEquation expressing the development of the alpha value during the animation.

GenericCallback < const FadeAnimator < T > &> * fadeAnimationEndedCallback
 Animation ended Callback.

7.76.1 Detailed Description

```
template < class T > class touchgfx::FadeAnimator < T >
```

A FadeAnimator makes the template class T able to animate an alpha fade from its current alpha value to a specified end alpha value. The alpha development can be described by supplying an EasingEquation. The FadeAnimator performs a callback when the animation has finished.

This mixin can be used on any Drawable that has a 'void setAlpha(uint8 t)' and a 'uint8 t getAlpha()' method.

Template Parameters

T | Specifies the type should have the fade animation capability.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 FadeAnimator()

```
FadeAnimator ( ) [inline]
```

Default constructor. Creates and initialize the FadeAnimator.

7.76.2.2 \sim FadeAnimator()

```
~FadeAnimator ( ) [inline], [virtual]
```

Destructor. Destroys the FadeAnimator.

7.76.3 Member Function Documentation

7.76.3.1 clearFadeAnimationEndedAction()

```
void clearFadeAnimationEndedAction ( ) [inline]
```

Clears the fade animation ended action previously set by setFadeAnimationEndedAction.

See also

setFadeAnimationEndedAction

7.76.3.2 getFadeAnimationDelay()

```
uint16_t getFadeAnimationDelay ( ) const [inline], [virtual]
```

Gets the current animation delay.

Returns

The current animation delay.

7.76.3.3 handleTickEvent()

```
void handleTickEvent ( ) [inline], [protected], [virtual]
```

The tick handler that handles the actual animation steps.

7.76.3.4 isFadeAnimationRunning()

```
bool isFadeAnimationRunning ( ) const [inline], [virtual]
```

Gets whether or not the fade animation is running.

Returns

true if the fade animation is running.

7.76.3.5 isRunning()

```
bool isRunning ( ) const [inline], [virtual]
```

Gets whether or not the fade animation is running.

Returns

true if the fade animation is running.

7.76.3.6 nextFadeAnimationStep()

```
void nextFadeAnimationStep ( ) [inline], [protected]
```

Execute next step in fade animation and stop the timer if necessary.

7.76.3.7 setFadeAnimationDelay()

Sets a delay on animations done by the FadeAnimator.

delay	The delay in ticks.

7.76.3.8 setFadeAnimationEndedAction()

Associates an action to be performed when the animation ends.

Parameters

callback	The callback to be executed.	The callback will be given a reference to the FadeAnimator.
----------	------------------------------	---

See also

GenericCallback

7.76.3.9 startFadeAnimation()

Starts the fade animation from the current alpha value to the specified end alpha value. The development of the alpha value during the animation is described by the supplied EasingEquation.

Parameters

endAlpha	The alpha value of T at animation end.
duration	The duration of the animation measured in ticks.
alphaProgressionEquation	The equation that describes the development of the alpha value during the animation. Default = EasingEquations::linearEaseNone.

7.77 FlashDataReader Class Reference

This class is an abstract interface for a class reading data from a flash.

```
#include <touchgfx/hal/FlashDataReader.hpp>
```

Public Member Functions

- virtual bool addresslsAddressable (const void *address)=0
 - Compute if an address is directly addressable by the MCU.
- virtual void copyData (const void *src, void *dst, uint32_t bytes)=0

Copy data from flash to a buffer.

- virtual void startFlashLineRead (const void *src, uint32_t bytes)
 - Initiate a read operation from flash to a buffer.
- virtual const uint8_t * waitFlashReadComplete ()

Waits until the previous startFlashLineRead operation is complete.

7.77.1 Detailed Description

This class is an abstract interface for a class reading data from a flash. The flash can be any type, but is mostly used for flashes that are not memory mapped. Applications must implement access to the flash through this interface.

7.77.2 Member Function Documentation

7.77.2.1 addressIsAddressable()

Compute if an address is directly addressable by the MCU. The data is addressable it should be read direct through a pointer and not through this interface.

Parameters

address	The address in the flash.
---------	---------------------------

Returns

True if the address is addressable by the MCU.

7.77.2.2 copyData()

Copy data from flash to a buffer. This must be a synchrony method that does not return until the copy is done.

Parameters

src	Address of source data in the flash
dst	Address of destination buffer in RAM
bytes	Number of bytes to copy.

7.77.2.3 startFlashLineRead()

Initiate a read operation from flash to a buffer. This can be an asynchrony operation that is still running after this function returns. Buffers must be handled by the subclass. LCD16bbbSerialFlash will at most copy 4 bytes times the width of the display.

Parameters

src	Address of source data in the flash
bytes	Number of bytes to copy.

7.77.2.4 waitFlashReadComplete()

```
const uint8_t * waitFlashReadComplete ( ) [virtual]
```

Waits until the previous startFlashLineRead operation is complete. If the startFlashLineRead method is asynchrony, this method must wait until the previous operation has completed.

Returns

The address of a buffer containing the read data.

7.78 Font Class Reference

The font base class.

```
#include <touchgfx/Font.hpp>
```

Public Member Functions

virtual const GlyphNode * getGlyph (Unicode::UnicodeChar unicode, const uint8_t *&pixelData, uint8_←
t &bitsPerPixel) const =0

Gets the glyph data associated with the specified unicode.

virtual const GlyphNode * getGlyph (Unicode::UnicodeChar unicode) const

Gets the glyph data associated with the specified unicode.

· virtual Unicode::UnicodeChar getFallbackChar () const

Gets fallback character.

virtual Unicode::UnicodeChar getEllipsisChar () const

Gets ellipsis character.

• virtual uint16_t getStringWidth (const Unicode::UnicodeChar *text,...) const

Gets the width in pixels of the specified string.

virtual uint16_t getStringWidth (TextDirection textDirection, const Unicode::UnicodeChar *text,...) const
 Gets the width in pixels of the specified string.

• virtual uint16_t getCharWidth (const Unicode::UnicodeChar c) const

Gets the width in pixels of the specified character.

virtual uint8_t getSpacingAbove (const Unicode::UnicodeChar *text,...) const

Gets the number of blank pixels at the top of the given text.

• virtual uint16_t getMaxTextHeight (const Unicode::UnicodeChar *text,...) const

Gets the height of the highest character in a given string.

· virtual uint16_t getFontHeight () const

Returns the height in pixels of this font.

virtual uint16_t getMinimumTextHeight () const

Returns the minimum height needed for a text field that uses this font.

virtual uint8_t getBitsPerPixel () const

Gets bits per pixel for this font.

virtual uint8_t getDataFormatA4 () const

7.78 Font Class Reference 345

Are the glyphs saved using ST A4 format.

uint8_t getMaxPixelsLeft () const

Gets maximum pixels left.

uint8_t getMaxPixelsRight () const

Gets maximum pixels right.

virtual int8_t getKerning (Unicode::UnicodeChar prevChar, const GlyphNode *glyph) const

Gets the kerning distance between two characters.

• virtual uint16_t getNumberOfLines (const Unicode::UnicodeChar *text,...) const

Gets number of lines.

virtual const uint16_t * getGSUBTable () const

Gets GSUB table.

Protected Types

typedef uint16_t(Font::* StringWidthFunctionPointer) (TextDirection textDirection, const Unicode::Unicode
 — Char *text, va_list pArg) const

Defines an alias representing the constant.

Protected Member Functions

uint16_t getStringWidthLTR (TextDirection textDirection, const Unicode::UnicodeChar *text, va_list pArg)
 const

Gets the width in pixels of the specified string.

uint16_t getStringWidthRTL (TextDirection textDirection, const Unicode::UnicodeChar *text, va_list pArg)
 const

Gets the width in pixels of the specified string.

• Font (uint16_t height, uint8_t pixBelowBase, uint8_t bitsPerPixel, uint8_t dataFormatA4, uint8_t maxLeft, uint8_t maxRight, const Unicode::UnicodeChar fallbackChar, const Unicode::UnicodeChar ellipsisChar)

Constructor.

Protected Attributes

· uint16 t fontHeight

The font height in pixels.

• uint8_t pixelsBelowBaseline

The number of pixels below the base line.

• uint8_t bPerPixel: 7

The number of bits per pixel.

• uint8_t a4: 1

Are glyphs encoded using A4 format.

uint8_t maxPixelsLeft

The maximum number of pixels a glyph extends to the left.

uint8_t maxPixelsRight

The maximum number of pixels a glyph extends to the right.

· Unicode::UnicodeChar falllbackCharacter

The fallback character to use when no glyph exists for the wanted character.

· Unicode::UnicodeChar ellipsisCharacter

The ellipsis character used for truncating long texts.

Static Protected Attributes

• static StringWidthFunctionPointer getStringWidthFunction

The getStringWidth function, either LTR (supporting LTR only) or RTL (supporting RTL and LTR)

7.78.1 Detailed Description

The font base class. This class is abstract and requires the implementation of getGlyph.

It provides utility functions such as obtaining string width and font height.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 Font()

The protected constructor of a Font.

Parameters

height	The font height in pixels.
pixBelowBase	The number of pixels below the base line.
bitsPerPixel	The number of bits per pixel.
dataFormatA4	The glyphs are saved using ST A4 format.
maxLeft	The maximum left extend for a glyph in the font.
maxRight	The maximum right extend for a glyph in the font.
fallbackChar	The fallback character for the typography in case no glyph is available.
ellipsisChar	The ellipsis character used for truncating long texts.

7.78.3 Member Function Documentation

```
7.78.3.1 getBitsPerPixel()
```

```
uint8_t getBitsPerPixel ( ) const [inline], [virtual]
```

Gets bits per pixel for this font.

7.78 Font Class Reference 347

Returns

The number of bits used per pixel in this font.

7.78.3.2 getCharWidth()

Gets the width in pixels of the specified character.

Parameters

```
c The unicode character.
```

Returns

The width in pixels of the specified character.

7.78.3.3 getDataFormatA4()

```
uint8_t getDataFormatA4 ( ) const [inline], [virtual]
```

Are the glyphs saved using ST A4 format.

Returns

True if the font is stored using A4 format, false otherwise.

7.78.3.4 getEllipsisChar()

```
Unicode::UnicodeChar getEllipsisChar ( ) const [inline], [virtual]
```

Gets ellipsis character for the given font. This is the character which is used when truncating long lines.

Returns

The ellipsis character for the typography.

7.78.3.5 getFallbackChar()

```
Unicode::UnicodeChar getFallbackChar ( ) const [inline], [virtual]
```

Gets fallback character for the given font. In case there is no glyph for a character, use the glyph for the character returned by this function. If 0 (zero) is returned, there is no default character.

Returns

The default character for the typography in case no glyph is available.

7.78.3.6 getFontHeight()

```
uint16_t getFontHeight ( ) const [inline], [virtual]
```

Returns the height in pixels of this font. The returned value corresponds to the maximum height occupied by a character in the font.

Note

It is not sufficient to allocate text areas with this height. Use getMinimumTextHeight for this.

Returns

The height in pixels of this font.

7.78.3.7 getGlyph() [1/2]

Gets the glyph data associated with the specified unicode. Please note that in case of Thai letters where diacritics can be placed relative to the previous character(s), please use TextProvider::getGlyph() instead as it will make a GlyphNode that will be correct with respect to X/Y position.

Parameters

		unicode	The character to look up.
		pixelData	Pointer to the pixel data for the glyph if the glyph is found. This is set by this method.
C	out	bitsPerPixel	Reference where to place the number of bits per pixel.

Returns

A pointer to the glyph node or null if the glyph was not found.

Implemented in ConstFont.

7.78.3.8 getGlyph() [2/2]

Gets the glyph data associated with the specified unicode. Please note that in case of Thai letters where diacritics can be placed relative to the previous character(s), please use TextProvider::getGlyph() instead as it will make a GlyphNode that will be correct with respect to X/Y position.

unicode The character to look up.

7.78 Font Class Reference 349

Returns

A pointer to the glyph node or null if the glyph was not found.

See also

TextProvider::getGlyph

7.78.3.9 getGSUBTable()

```
uint16_t * getGSUBTable ( ) const [inline], [virtual]
```

Gets GSUB table.

Returns

The GSUB table or null if font has GSUB no table

7.78.3.10 getKerning()

Gets the kerning distance between two characters.

Parameters

prevChar	The unicode value of the previous character.
glyph	the glyph object for the current character.

Returns

The kerning distance between prevChar and glyph char.

Reimplemented in ConstFont, and InternalFlashFont.

7.78.3.11 getMaxPixelsLeft()

```
uint8_t getMaxPixelsLeft ( ) const [inline]
```

Gets maximum pixels left for any glyph in the font. This is the max value of "left" for all glyphs. The value is negated so if a "g" has left=-6 maxPixelsLeft is 6. This value is calculated by the font converter.

Returns

The maximum pixels left.

7.78.3.12 getMaxPixelsRight()

```
uint8_t getMaxPixelsRight ( ) const [inline]
```

Gets maximum pixels right for any glyph in the font. This is the max value of "width+left-advance" for all glyphs. The is the number of pixels a glyph reaches to the right of its normal area. This value is calculated by the font converter.

Returns

The maximum pixels right.

7.78.3.13 getMaxTextHeight()

Gets the height of the highest character in a given string. The height includes the spacing above the text which is included in the font.

Parameters

text	A zero-terminated unicode string.
	Variable arguments providing additional information.

Returns

The height if the given text.

7.78.3.14 getMinimumTextHeight()

```
uint16_t getMinimumTextHeight ( ) const [inline], [virtual]
```

Returns the minimum height needed for a text field that uses this font. Takes into account that certain characters (eg 'g') have pixels below the baseline, thus making the text height larger than the font height.

Returns

The minimum height needed for a text field that uses this font.

7.78.3.15 getNumberOfLines()

Count the number of lines in a given text.

text	The text.	
	Variable arguments providing additional information.	

7.78 Font Class Reference 351

Returns

The number of lines.

7.78.3.16 getSpacingAbove()

Gets the number of blank pixels at the top of the given text.

Parameters

text	A zero-terminated unicode string.
	Variable arguments providing additional information.

Returns

The number of blank pixels above the text.

Gets the width in pixels of the specified string. If the string contains multiple lines, the width of the widest line is found. Please note that the correct number of arguments must be given if the text contains wildcards.

It is recommended to use the getStringWidth() implementation with the TextDirection parameter to ensure correct calculation of the width. Kerning could result in different results depending on the TextDirection. This method assumes TextDirection to be TEXT_DIRECTION_LTR.

Parameters

tex	A zero-terminated unicode string with arguments to insert if the text contains wildcards.
	Variable arguments providing additional information.

Returns

The width in pixels of the longest line of the specified string.

Gets the width in pixels of the specified string. If the string contains multiple lines, the width of the widest line is found. Please note that the correct number of arguments must be given if the text contains wildcards.

The TextDirection should be set correctly for the text supplied. For example the string "10 20 30" will be calculated differently depending on the TextDirection. If TextDirection is TEXT_DIRECTION_LTR the width is calculated as the with of "10 20 30" (with kerning between all characters) but for TEXT_DIRECTION_RTL it is calculated as "10"+" "+"20"+" "+"30" (with kerning only between characters in the substrings and not between substrings). For most fonts there might not be a difference between the two calculations, but some fonts might cause different results.

Parameters

textDirection	The text direction.
text	A zero-terminated unicode string with arguments to insert if the text contains wildcards.
	Variable arguments providing additional information.

Returns

The width in pixels of the longest line of the specified string.

7.78.3.19 getStringWidthLTR()

Gets the width in pixels of the specified string. If the string contains multiple lines, the width of the widest line is found. Please note that the correct number of arguments must be given if the text contains wildcards.

The string is assumed to be purely left-to-right.

Parameters

textDirection	The text direction.
text	A zero-terminated unicode string with arguments to insert if the text contains wildcards.
pArg	Variable arguments providing additional information.

Returns

The width in pixels of the longest line of the specified string.

7.78.3.20 getStringWidthRTL()

Gets the width in pixels of the specified string. If the string contains multiple lines, the width of the widest line is found. Please note that the correct number of arguments must be given if the text contains wildcards.

The string is handled as a right-to-left string and subdivided into smaller text strings to correctly handle mixing of left-to-right and right-to-left strings.

textDirection	The text direction.

Parameters

text	A zero-terminated unicode string with arguments to insert if the text contains wildcards.
pArg	The argument.

Returns

The string width RTL.

7.79 FontManager Class Reference

This class is the entry point for looking up a font based on a font id.

```
#include <touchgfx/FontManager.hpp>
```

Static Public Member Functions

- $\bullet \ \ \mathsf{static} \ \mathsf{void} \ \mathsf{setFontProvider} \ (\mathsf{FontProvider} \ *\mathsf{fontProvider}) \\$
 - Sets the font provider.
- static Font * getFont (FontId fontId)

Gets a font.

7.79.1 Detailed Description

This class is the entry point for looking up a font based on a font id. Must be initialized with the appropriate Font← Provider by the application.

7.79.2 Member Function Documentation

7.79.2.1 getFont()

Gets a font.

Parameters

font⇔	The font id of the font to get.
ld	

Returns

The font with a font id of fontld.

7.79.2.2 setFontProvider()

Sets the font provider. Must be initialized with the appropriate FontProvider by the application.

Parameters

in	fontProvider	Sets the font provider. Must be initialized with the appropriate FontProvider by the application.
----	--------------	---

7.80 FontProvider Class Reference

A generic pure virtual definition of a FontProvider.

```
#include <touchgfx/FontManager.hpp>
```

Public Member Functions

virtual Font * getFont (FontId fontId)=0
 Gets a font.

virtual ∼FontProvider ()

Destructor.

7.80.1 Detailed Description

A generic pure virtual definition of a FontProvider, which is a class capable of returning a font based on a font id. An application-specific derivation of this class must be implemented.

7.80.2 Constructor & Destructor Documentation

```
7.80.2.1 ~FontProvider()
~FontProvider ( ) [inline], [virtual]
Destructor.
```

7.80.3 Member Function Documentation

Gets a font.

font⇔	The font id of the font to get.
ld	

Returns

The font with a font id of fontld.

7.81 FrameBufferAllocator Class Reference

This class is an abstract interface for a class allocating partial framebuffer blocks.

```
#include <touchgfx/hal/FrameBufferAllocator.hpp>
```

Public Member Functions

 virtual uint16_t allocateBlock (const uint16_t x, const uint16_t y, const uint16_t width, const uint16_t height, uint8_t **block)=0

Allocates a framebuffer block.

virtual void markBlockReadyForTransfer ()=0

Marks a previously allocated block as ready to be transferred to the LCD.

• virtual bool hasBlockReadyForTransfer ()=0

Check if a block is ready for transfer to the LCD.

virtual const uint8_t * getBlockForTransfer (Rect &rect)=0

Get the block ready for transfer.

virtual void freeBlockAfterTransfer ()=0

Free a block after transfer to the LCD.

7.81.1 Detailed Description

This class is an abstract interface for a class allocating partial framebuffer blocks. The interface must be implemented by a subclass.

See also

SingleBlockAllocator, ManyBlockAllocator

7.81.2 Member Function Documentation

7.81.2.1 allocateBlock()

Allocates a framebuffer block. The block will have at least the width requested. The height of the allocated block can be lower than requested if not enough memory is available.

X	The absolute x coordinate of the block on the screen.
У	The absolute y coordinate of the block on the screen.
width	The width of the block.

Parameters

height	The height of the block.
block	Pointer to pointer to return the block address in.

Returns

The height of the allocated block.

Implemented in ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >, and SingleBlockAllocator< block_size, bytes_pr_pixel >.

7.81.2.2 freeBlockAfterTransfer()

```
void freeBlockAfterTransfer ( ) [pure virtual]
```

Marks a previously allocated block as transferred and ready to reuse.

Implemented in ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >, and SingleBlockAllocator< block_size, bytes_pr_pixel >.

7.81.2.3 getBlockForTransfer()

Get the block ready for transfer.

Parameters

rect	Reference to rect to write block x, y, width, and height.
------	---

Returns

Returns the address of the block ready for transfer.

Implemented in ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >, and SingleBlockAllocator< block_size, bytes pr pixel >.

7.81.2.4 hasBlockReadyForTransfer()

```
bool hasBlockReadyForTransfer ( ) [pure virtual]
```

Check if a block is ready for transfer to the LCD.

Returns

True if a block is ready for transfer.

Implemented in ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >, and SingleBlockAllocator< block_size, bytes_pr_pixel >.

7.81.2.5 markBlockReadyForTransfer()

```
void markBlockReadyForTransfer ( ) [pure virtual]
```

Marks a previously allocated block as ready to be transferred to the LCD.

Implemented in ManyBlockAllocator< block_size, blocks, bytes_pr_pixel >, and SingleBlockAllocator< block_size, bytes_pr_pixel >.

7.82 CoverTransition < templateDirection >::FullSolidRect Class Reference

A Widget that returns a solid rect of the same size as the application.

```
#include <CoverTransition.hpp>
```

Public Member Functions

· virtual Rect getSolidRect () const

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent).

· virtual void draw (const Rect &area) const

Pure virtual function for drawing this drawable.

Additional Inherited Members

7.82.1 Member Function Documentation

7.82.1.1 draw()

Pure virtual function for drawing this drawable. It is a requirement that the draw implementation does not draw outside the region specified by invalidatedArea.

Parameters

invalidatedArea	The subregion of this drawable that needs to be redrawn, expressed in coordinates relative
	to its parent (e.g. for a complete redraw, invalidatedArea will be (0, 0, width, height).

Implements Drawable.

7.82.1.2 getSolidRect()

```
virtual Rect getSolidRect ( ) const [inline], [virtual]
```

Pure virtual function for obtaining the largest possible rectangle that is guaranteed to be solid (non-transparent). Used by JSMOC to prune the draw graph.

Note

The rectangle returned must be relative to (0, 0), meaning that to indicate a completely solid widget, Rect(0, 0, getWidth(), getHeight()) must be returned.

Returns

The solid rect.

Implements Drawable.

7.83 GenericCallback< T1, T2, T3 > Class Template Reference

GenericCallback is the base class for callbacks.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

virtual ∼GenericCallback ()

Destructor.

virtual void execute (T1 val1, T2 val2, T3 val3)=0

Calls the member function.

virtual bool isValid () const =0

Function to check whether the Callback has been initialized with values.

7.83.1 Detailed Description

```
template<class T1 = void, class T2 = void, class T3 = void> class touchgfx::GenericCallback< T1, T2, T3 >
```

GenericCallback is the base class for callbacks.

See also

Callback for an explanation of callbacks.

The reason this base class exists, is that a normal Callback requires the class type where the callback function resides to be known. This is problematic for ie. framework widgets like AbstractButton, on which it should be possible to register a callback on object types that are user-specific and thus unknown to AbstractButton. This is solved by having AbstractButton contain a pointer to a GenericCallback instead. This pointer must then be initialized to point on an instance of Callback, created by the user, which is initialized with the appropriate object type.

Note

As with Callback, this class exists in four versions to support callback functions taking zero, one, two or three arguments.

Template Parameters

T1	The type of the first argument in the member function, or void if none.
T2	The type of the second argument in the member function, or void if none.
Т3	The type of the third argument in the member function, or void if none.

7.83.2 Constructor & Destructor Documentation

7.83.2.1 ∼GenericCallback()

```
~GenericCallback ( ) [inline], [virtual]
```

Empty virtual destructor.

7.83.3 Member Function Documentation

7.83.3.1 execute()

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

val1	This value will be passed as the first argument in the function call.
val2	This value will be passed as the second argument in the function call.
val3	This value will be passed as the third argument in the function call.

Implemented in Callback< dest_type, T1, T2, T3 >, Callback< touchgfx::SlideTransition, touchgfx::Drawable &>, Callback< touchgfx::RadioButtonGroup, const touchgfx::AbstractButton & >, Callback< touchgfx::CoverTransition, touchgfx::Drawable &>, Callback< touchgfx::SlideMenu, const touchgfx::AbstractButton &>, and Callback< touchgfx::SlideMenu, const touchgfx::MoveAnimator< touchgfx::Container > &>.

7.83.3.2 isValid()

```
bool isValid ( ) const [pure virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

Implemented in Callback< dest_type, void, void >, Callback< dest_type, T1, void, void >, Callback< dest_type, T1, void, void >, Callback< dest_type, T1, T2, void >, Callback< dest_type, T1, T2, T3 >, Callback< touchgfx::GlideTransition, touchgfx::GlideTransition, touchgfx::GlideTransition, touchgfx::AbstractButton & >, Callback< touchgfx::GlideTransition, touchgfx::Drawable & >, Callback< touchgfx::SlideMenu, const touchgfx::AbstractButton & >, and Callback< touchgfx::SlideMenu, const touchgfx::Gontainer > & >.

7.84 GenericCallback< T1, T2, void > Class Template Reference

GenericCallback is the base class for callbacks.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

virtual ∼GenericCallback ()

Destructor.

• virtual void execute (T1 val1, T2 val2)=0

Calls the member function.

• virtual bool isValid () const =0

Function to check whether the Callback has been initialized with values.

7.84.1 Detailed Description

```
\label{template} \begin{tabular}{ll} template < class T1, class T2 > \\ class touchgfx::GenericCallback < T1, T2, void > \\ \end{tabular}
```

GenericCallback is the base class for callbacks.

See also

Callback for an explanation of callbacks.

The reason this base class exists, is that a normal Callback requires the class type where the callback function resides to be known. This is problematic for ie. framework widgets like AbstractButton, on which it should be possible to register a callback on object types that are user-specific and thus unknown to AbstractButton. This is solved by having AbstractButton contain a pointer to a GenericCallback instead. This pointer must then be initialized to point on an instance of Callback, created by the user, which is initialized with the appropriate object type.

Note

As with Callback, this class exists in four versions to support callback functions taking zero, one, two or three arguments.

Template Parameters

7	1	The type of the first argument in the member function, or void if none.
7	2	The type of the second argument in the member function, or void if none.

7.84.2 Constructor & Destructor Documentation

```
7.84.2.1 ~GenericCallback() ~GenericCallback ( ) [inline], [virtual] Empty virtual destructor.
```

7.84.3 Member Function Documentation

7.84.3.1 execute()

```
virtual void execute (  \begin{tabular}{ll} $T1$ $val1,$\\ $T2$ $val2$ ) [pure virtual] \end{tabular}
```

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

val1	This value will be passed as the first argument in the function call.
val2	This value will be passed as the second argument in the function call.

7.84.3.2 isValid()

```
bool isValid ( ) const [pure virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

7.85 GenericCallback< T1, void, void > Class Template Reference

GenericCallback is the base class for callbacks.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

virtual ∼GenericCallback ()

Destructor.

• virtual void execute (T1 val1)=0

Calls the member function.

virtual bool isValid () const =0

Function to check whether the Callback has been initialized with values.

7.85.1 Detailed Description

```
\label{template} $$ $$ $$ template < class T1>$$ $$ class touchgfx::GenericCallback < T1, void, void > $$
```

GenericCallback is the base class for callbacks.

See also

Callback for an explanation of callbacks.

The reason this base class exists, is that a normal Callback requires the class type where the callback function resides to be known. This is problematic for ie. framework widgets like AbstractButton, on which it should be possible to register a callback on object types that are user-specific and thus unknown to AbstractButton. This is solved by having AbstractButton contain a pointer to a GenericCallback instead. This pointer must then be initialized to point on an instance of Callback, created by the user, which is initialized with the appropriate object type.

Note

As with Callback, this class exists in four versions to support callback functions taking zero, one, two or three arguments.

Template Parameters

T1 The type of the first argument in the member function, or void if none.

7.85.2 Constructor & Destructor Documentation

```
7.85.2.1 ∼GenericCallback()
```

```
~GenericCallback () [inline], [virtual]
```

Empty virtual destructor.

7.85.3 Member Function Documentation

7.85.3.1 execute()

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

Parameters

val1 This value will be passed as the first argument in the function call.

7.85.3.2 isValid()

```
bool isValid ( ) const [pure virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

7.86 GenericCallback < void > Class Template Reference

GenericCallback is the base class for callbacks.

```
#include <touchgfx/Callback.hpp>
```

Public Member Functions

virtual ∼GenericCallback ()

Destructor.

• virtual void execute ()=0

Calls the member function.

• virtual bool isValid () const =0

Function to check whether the Callback has been initialized with values.

7.86.1 Detailed Description

template<> class touchgfx::GenericCallback< void >

GenericCallback is the base class for callbacks.

See also

Callback for an explanation of callbacks.

The reason this base class exists, is that a normal Callback requires the class type where the callback function resides to be known. This is problematic for ie. framework widgets like AbstractButton, on which it should be possible to register a callback on object types that are user-specific and thus unknown to AbstractButton. This is solved by having AbstractButton contain a pointer to a GenericCallback instead. This pointer must then be initialized to point on an instance of Callback, created by the user, which is initialized with the appropriate object type.

Note

As with Callback, this class exists in four versions to support callback functions taking zero, one, two or three arguments.

7.86.2 Constructor & Destructor Documentation

```
7.86.2.1 ~GenericCallback()
  ~GenericCallback ( ) [inline], [virtual]
Empty virtual destructor.
```

7.86.3 Member Function Documentation

```
7.86.3.1 execute()
void execute ( ) [pure virtual]
```

Calls the member function. Do not call execute unless is Valid() returns true (ie. a pointer to the object and the function has been set).

7.86.3.2 isValid()

```
bool isValid ( ) const [pure virtual]
```

Function to check whether the Callback has been initialized with values.

Returns

true If the callback is valid (i.e. safe to call execute).

7.87 GestureEvent Class Reference

A gesture event.

```
#include <touchgfx/events/GestureEvent.hpp>
```

Public Types

enum GestureType { SWIPE_HORIZONTAL, SWIPE_VERTICAL }

The gesture event types.

Public Member Functions

GestureEvent (GestureType t, int16_t v, int16_t x_coord, int16_t y_coord)

Constructor.

• int16_t getVelocity () const

Gets the velocity of this gesture event.

GestureType getType () const

Gets the type of this gesture event.

int16_t getX () const

Gets the x coordinate of this gesture event.

int16_t getY () const

Gets the y coordinate of this gesture event.

virtual Event::EventType getEventType ()

Gets event type.

7.87.1 Detailed Description

A gesture event. The only gesture events currently supported is SWIPE_HORIZONTAL and SWIPE_VERTICAL, which will be issued every time the input system detects a swipe.

See also

Event

7.87.2 Member Enumeration Documentation

7.87.2.1 GestureType

enum enum GestureType

Enumerator

SWIPE_HORIZONTAL	An enum constant representing a horizontal swipe.
SWIPE_VERTICAL	An enum constant representing a vertical swipe.

7.87.3 Constructor & Destructor Documentation

7.87.3.1 GestureEvent()

```
GestureEvent (
          GestureType t,
          int16_t v,
          int16_t x_coord,
          int16_t y_coord ) [inline]
```

Constructor. Create a gesture event of the specified type with the specified coordinates.

Parameters

t	The type of the gesture event.
V	The velocity of this gesture (swipe)
x_coord	The x coordinate of the gesture.
y_coord	The y coordinate of the gesture.

7.87.4 Member Function Documentation

7.87.4.1 getEventType()

```
Event::EventType getEventType ( ) [inline], [virtual]
```

Returns

The type of this event.

Implements Event.

7.87.4.2 getType()

```
GestureType getType ( ) const [inline]
```

Gets the type of this gesture event.

Returns

The type of this gesture event.

7.87.4.3 getVelocity()

```
int16_t getVelocity ( ) const [inline]
```

Gets the velocity of this gesture event.

Returns

The velocity of this gesture event.

7.87.4.4 getX()

```
int16_t getX ( ) const [inline]
```

Gets the x coordinate of this gesture event.

Returns

The x coordinate of this gesture event.

7.87.4.5 getY()

```
int16_t getY ( ) const [inline]
```

Gets the y coordinate of this gesture event.

Returns

The y coordinate of this gesture event.

7.88 Gestures Class Reference

This class implements the detection of gestures.

```
#include <touchgfx/hal/Gestures.hpp>
```

Public Member Functions

· Gestures ()

Default constructor.

void registerEventListener (UIEventListener &I)

Register the event listener.

• void tick ()

Has to be called during the timer tick.

• bool registerDragEvent (uint16_t oldX, uint16_t oldY, uint16_t newX, uint16_t newY)

Register a drag event.

void registerClickEvent (ClickEvent::ClickEventType evt, uint16_t x, uint16_t y)

Register a click event and figure out if this is a drag event, too.

void setDragThreshold (uint16_t val)

Configure the threshold for reporting drag events.

7.88.1 Detailed Description

This class implements the detection of gestures.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 Gestures()

```
Gestures ( ) [inline]
```

Default constructor. Does nothing.

7.88.3 Member Function Documentation

7.88.3.1 registerClickEvent()

Register a click event and figure out if this is a drag event, too.

Parameters

evt	The type of the click event.
Х	The x coordinate of the click event.
У	The y coordinate of the click event.

7.88.3.2 registerDragEvent()

Register a drag event.

Talamotoro		
oldX	The x coordinate of the drag start position (dragged from)	
oldY	The y coordinate of the drag start position (dragged from)	
newX	The x coordinate of the new position (dragged to)	
newY	The y coordinate of the new position (dragged to)	

Returns

True if the drag exceeds threshold value (and therefore was reported as a drag), or false if the drag did not exceed threshold (and therefore was discarded).

7.88.3.3 registerEventListener()

```
void register
EventListener ( {\tt UIEventListener~\&~l~)}
```

Register the event listener.

Parameters

in / The EventListener to reg	jister.
-------------------------------	---------

7.88.3.4 setDragThreshold()

Configure the threshold for reporting drag events. A touch input movement must exceed this value in either axis in order to report a drag. Default value is 0.

Parameters

```
val New threshold value.
```

7.88.3.5 tick()

```
void tick ( )
```

Has to be called during the timer tick.

7.89 GlyphNode Struct Reference

struct providing information about a glyph.

```
#include <touchgfx/Font.hpp>
```

Public Member Functions

• uint16_t kerningTablePos () const

Gets the "kerningTablePos" value where the 8th and 9th bits is stored in flags.

• uint16_t width () const

Gets the "width" value where the 9th bit is stored in flags.

• uint16_t height () const

Gets the "height" value where the 9th bit is stored in flags.

• int16_t top () const

Gets the "top" value where the 9th bit and the sign bit are stored in flags.

void setTop (int16_t newTop)

Sets a top.

• uint16_t advance () const

Gets the "advance" value where the 9th bit is stored in flags.

Public Attributes

uint32_t dataOffset

The index to the data of this glyph.

· Unicode::UnicodeChar unicode

The unicode of this glyph.

· uint8_t _width

Width of the actual glyph data.

uint8_t _height

Height of the actual glyph data.

uint8_t _top

Vertical offset from baseline of the glyph.

int8_t left

Horizontal offset from the left of the glyph.

· uint8 t advance

Width of the glyph (including space to the left and right)

uint8_t _kerningTablePos

Where are the kerning information for this glyph stored in the kerning table.

• uint8_t kerningTableSize

How many entries are there in the kerning table (following kerningTablePos) for this glyph.

• uint8_t flags

Additional glyph flags (font encoding and extra precision for width/height/top/advance)

7.89.1 Detailed Description

struct providing information about a glyph. Used by LCD when rendering.

7.89.2 Member Function Documentation

```
7.89.2.1 advance()
```

```
uint16_t advance ( ) const [inline]
```

Gets the "advance" value where the 9th bit is stored in flags.

Returns

the right value of "advance".

7.89.2.2 height()

```
uint16_t height ( ) const [inline]
```

Gets the "height" value where the 9th bit is stored in flags.

Returns

the right value of "height".

7.89.2.3 kerningTablePos()

```
uint16_t kerningTablePos ( ) const [inline]
```

Gets the "kerningTablePos" value where the 8th and 9th bits is stored in flags.

Returns

the right value of "kerningTablePos".

7.89.2.4 setTop()

```
void setTop (
          int16_t newTop ) [inline]
```

Parameters

newTop	The new top.
--------	--------------

7.89.2.5 top()

```
int16_t top ( ) const [inline]
```

Gets the "top" value where the 9th bit and the sign bit are stored in flags.

Returns

the right value of "top".

7.89.2.6 width()

```
uint16_t width ( ) const [inline]
```

Gets the "width" value where the 9th bit is stored in flags.

Returns

the right value of "width".

7.90 GPIO Class Reference 371

7.90 GPIO Class Reference

Interface class for manipulating GPIOs in order to do performance measurements on target.

```
#include <touchgfx/hal/GPIO.hpp>
```

Public Types

enum GPIO_ID { VSYNC_FREQ, RENDER_TIME, FRAME_RATE, MCU_ACTIVE }
 Enum for the GPIOs used.

Static Public Member Functions

• static void init ()

Perform configuration of IO pins.

• static void set (GPIO_ID id)

Sets a pin high.

static void clear (GPIO_ID id)

Sets a pin low.

• static void toggle (GPIO_ID id)

Toggles a pin.

static bool get (GPIO_ID id)

Gets the state of a pin.

7.90.1 Detailed Description

Interface class for manipulating GPIOs in order to do performance measurements on target. Not used on the PC simulator.

7.90.2 Member Enumeration Documentation

7.90.2.1 GPIO_ID

enum enum GPIO_ID

Enum for the GPIOs used.

Enumerator

RENDER_TIME	Pin is toggled at each VSYNC.	
FRAME_RATE	Pin is high when frame rendering begins, low when finished.	
MCU_ACTIVE Pin is toggled when the frame buffers are swapped. Pin is high when the MCU is doing work (i.e. not in idle task).		

7.90.3 Member Function Documentation

```
7.90.3.1 clear()
```

Sets a pin low.

Parameters

id the pin to set.

```
7.90.3.2 get()
```

Gets the state of a pin.

Parameters

```
id the pin to get.
```

Returns

true if the pin is high, false otherwise.

```
7.90.3.3 init()
```

```
static void init ( ) [static]
```

Perform configuration of IO pins.

```
7.90.3.4 set()
```

Sets a pin high.

Parameters

```
id the pin to set.
```

7.90.3.5 toggle()

Toggles a pin.

Parameters

id the pin to toggle.

7.91 Gradients Struct Reference

Gradients contains all the data to interpolate u,v texture coordinates and z coordinates across a planar surface.

```
#include <touchgfx/TextureMapTypes.hpp>
```

Public Member Functions

• Gradients (const Point3D *vertices)

Constructor. Construct the gradients using 3 3D vertices.

Public Attributes

• float oneOverZ [3]

1/z for each vertex

• float UOverZ [3]

u/z for each vertex

float VOverZ [3]

v/z for each vertex

float dOneOverZdX

d(1/z)/dX

float dOneOverZdY

d(1/z)/dY

float dUOverZdX

d(u/z)/dX

float dUOverZdY

d(u/z)/dY

float dVOverZdX

d(v/z)/dX

float dVOverZdY

d(v/z)/dY

• fixed16_16 dUdXModifier

The dUdX x coordinate modifier.

· fixed16 16 dVdXModifier

The dVdX x coordinate modifier.

7.91.1 Constructor & Destructor Documentation

7.91.1.1 Gradients()

Parameters

vertices	The vertices.
----------	---------------

See also

Point3D

7.92 HAL Class Reference

Hardware Abstraction Layer.

```
#include <touchgfx/hal/HAL.hpp>
```

Public Types

 enum FrameRefreshStrategy { REFRESH_STRATEGY_DEFAULT, REFRESH_STRATEGY_OPTIM_SIN← GLE_BUFFER_TFT_CTRL, REFRESH_STRATEGY_PARTIAL_FRAMEBUFFER }

A list of available frame refresh strategies.

Public Member Functions

HAL (DMA_Interface &dmaInterface, LCD &display, TouchController &touchCtrl, uint16_t width, uint16_
 t height)

Creates a HAL instance.

virtual ∼HAL ()

Destructor.

virtual void setDisplayOrientation (DisplayOrientation orientation)

Sets the desired display orientation (landscape or portrait).

· DisplayOrientation getDisplayOrientation () const

Gets the current display orientation.

void signalDMAInterrupt ()

Notify the framework that a DMA interrupt has occurred.

• void initialize ()

This function is responsible for initializing the entire framework.

virtual void taskEntry ()

Main event loop.

virtual void flushFrameBuffer ()

This function is called whenever the framework has performed a complete draw.

virtual void flushFrameBuffer (const Rect &rect)

This function is called whenever the framework has performed a partial draw.

· virtual void allowDMATransfers ()

Allow the DMA to start transfers.

• void frontPorchEntered ()

Has to be called from within the LCD IRQ routine when the Front Porch Entry is reached.

virtual void flushDMA ()

This function blocks until the DMA queue (containing BlitOps) is empty.

virtual uint16 t * lockFrameBuffer ()

Waits for the frame buffer to become available for use.

virtual void unlockFrameBuffer ()

Unlocks the frame buffer (MUST be called exactly once for each call to lockFrameBuffer()).

7.92 HAL Class Reference 375

virtual uint16_t * getTFTFrameBuffer () const =0

Gets the frame buffer address used by the TFT controller.

void lockDMAToFrontPorch (bool enableLock)

Function to set whether the DMA transfers are locked to the TFT update cycle.

virtual void registerTextCache (Unicode::UnicodeChar *str, uint16_t length)

Configures HAL to use the supplied buffer as text string cache.

virtual const Unicode::UnicodeChar * cacheTextString (const Unicode::UnicodeChar *str)

This function can be used to cache a given string.

virtual bool blockCopy (void *RESTRICT dest, const void *RESTRICT src, uint32 t numBytes)

This function performs a platform-specific memcpy.

virtual BlitOperations getBlitCaps ()

Function for obtaining the blit capabilities of the concrete HAL implementation.

virtual void blitSetTransparencyKey (uint16_t key)

Deprecated function which can be ignored.

Blits a 2D source-array to the frame buffer performing alpha-blending.

virtual void blitCopy (const uint16_t *pSrc, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_←
t srcWidth, uint8_t alpha, bool hasTransparentPixels, uint16_t dstWidth, Bitmap::BitmapFormat srcFormat,
Bitmap::BitmapFormat dstFormat)

Blits a 2D source-array to the frame buffer performing alpha-blending.

• virtual void blitCopy (const uint16_t *pSrc, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t srcWidth, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the frame buffer performing alpha-blending.

virtual void blitCopyARGB8888 (const uint16_t *pSrc, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t srcWidth, uint8_t alpha)

Blits a 2D source-array to the frame buffer performing per-pixel alpha blending.

• virtual void blitCopyGlyph (const uint8_t *pSrc, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint16_t srcWidth, colortype color, uint8_t alpha, BlitOperations operation)

Blits a 4bpp or 8bpp glyph - maybe use the same method and supply additional color mode arg.

virtual void blitFill (colortype color, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint8_t alpha, uint16

_t dstWidth, Bitmap::BitmapFormat dstFormat)

Blits a color value to the frame buffer performing alpha-blending (and transparency keying) as specified.

virtual void blitFill (colortype color, uint16_t x, uint16_t y, uint16_t width, uint16_t height, uint8_t alpha)

Blits a color value to the frame buffer performing alpha-blending (and transparency keying) as specified.

· virtual void registerEventListener (UIEventListener &listener)

Registers an event handler implementation with the underlying event system.

virtual uint16_t * copyFBRegionToMemory (Rect meAbs)

Copies a region of the currently displayed frame buffer to memory.

virtual uint16_t * copyFBRegionToMemory (Rect meAbs, uint16_t *dst, uint32_t stride)

Copies a region of the currently displayed frame buffer to memory.

• uint16 t getDisplayWidth () const

Gets display width.

uint16_t getDisplayHeight () const

Gets display height.

void swapFrameBuffers ()

Swaps the two frame buffers.

uint32_t getLCDRefreshCount ()

Returns the number of VSync interrupts.

void setFrameRateCompensation (bool enabled)

Enables or disables compensation for lost frames.

• void vSync ()

Called by the VSync interrupt.

virtual void backPorchExited ()

Has to be called from within the LCD IRQ rutine when the Back Porch Exit is reached.

virtual void configureInterrupts ()=0

Configures the interrupts relevant for TouchGFX.

• virtual void enableInterrupts ()=0

Enables the DMA and LCD interrupts.

virtual void disableInterrupts ()=0

Disables the DMA and LCD interrupts.

• virtual void enableLCDControllerInterrupt ()=0

Configure the LCD controller to fire interrupts at VSYNC.

virtual bool sampleKey (uint8_t &key)

Sample external key event.

void setDragThreshold (uint8_t value)

Configure the threshold for reporting drag events.

• virtual void setFrameBufferStartAddress (void *adr, uint16_t depth=16, bool useDoubleBuffering=true, bool useAnimationStorage=true)

Sets the address used for frame buffers, usually located in external memory.

virtual void setFrameBufferStartAddresses (void *frameBuffer, void *doubleBuffer, void *animationStorage)

Sets frame buffer start addresses.

 virtual uint16_t configurePartialFrameBuffer (const uint16_t x, const uint16_t y, const uint16_t width, const uint16 t height)

Configures a partial framebuffer as current framebuffer.

void setTouchSampleRate (int8_t sampleRateInTicks)

Sets the number of ticks between each touch screen sample.

• int8_t getTouchSampleRate () const

Gets the number of ticks between each touch screen sample.

void setMCUActive (bool active)

Register if MCU is active by measuring cpu cycles.

uint32_t getCPUCycles ()

Gets the current cycle counter.

void setMCUInstrumentation (MCUInstrumentation *mcuInstr)

Stores a pointer to an instance of an MCU specific instrumentation class.

• void enableMCULoadCalculation (bool enabled)

This method sets a flag that determines if generic HAL should calculate MCU load.

uint8_t getMCULoadPct () const

Gets the current MCU load.

void setButtonController (ButtonController *btnCtrl)

Stores a pointer to an instance of a specific implementation of a ButtonController.

• ButtonController * getButtonController () const

Gets the ButtonController.

void setFrameBufferAllocator (FrameBufferAllocator *allocator)

Sets a framebuffer allocator.

FrameBufferAllocator * getFrameBufferAllocator ()

Gets the framebuffer allocator.

void setFingerSize (uint8_t size)

Sets the finger size.

• uint8 t getFingerSize () const

Gets the finger size.

uint16_t * getAnimationStorage () const

7.92 HAL Class Reference 377

Gets the optional frame buffer used for animation storage.

bool setFrameRefreshStrategy (FrameRefreshStrategy s)

Set a specific strategy for handling timing and mechanism of frame buffer drawing.

• FrameRefreshStrategy getFrameRefreshStrategy () const

Used internally by TouchGFX core to manage the timing and process of drawing into the frame buffer.

void registerTaskDelayFunction (void(*delayF)(uint16_t))

Registers a function capable of delaying GUI task execution.

virtual void taskDelay (uint16_t ms)

Delay GUI task execution by number of milliseconds.

• virtual uint16_t getTFTCurrentLine ()

Get the current line (Y) of the TFT controller.

virtual DMAType getDMAType ()

Function for obtaining the DMA type of the concrete DMA implementation.

· virtual void drawDrawableInDynamicBitmap (Drawable &drawable, BitmapId bitmapId)

Render a Drawable and its widgets into a dynamic bitmap.

• virtual void drawDrawableInDynamicBitmap (Drawable &drawable, BitmapId bitmapId, const Rect &rect)

Render a Drawable and its widgets into a dynamic bitmap.

void setAuxiliaryLCD (LCD *auxLCD)

Set an auxiliary LCD class to be used for offscreen rendering.

LCD * getAuxiliaryLCD ()

Get the auxiliary LCD class attached to the HAL instance if any.

Static Public Member Functions

• static HAL * getInstance ()

Gets the HAL instance.

• static LCD & lcd ()

Gets a reference to the LCD.

Static Public Attributes

static uint16_t DISPLAY_WIDTH

The width of the LCD display in pixels.

static uint16_t DISPLAY_HEIGHT

The height of the LCD display in pixels.

static DisplayRotation DISPLAY_ROTATION

The rotation from display to frame buffer.

static uint16_t FRAME_BUFFER_WIDTH

The width of the frame buffer in pixels.

static uint16_t FRAME_BUFFER_HEIGHT

The height of the frame buffer in pixels.

• static bool USE_DOUBLE_BUFFERING

Is double buffering enabled?

static bool USE ANIMATION STORAGE

Is animation storage enabled?

Protected Member Functions

· virtual void tick ()

This function is called at each timer tick, depending on platform implementation.

virtual bool beginFrame ()

Called when beginning to rendering a frame.

virtual void endFrame ()

Called when a rendering pass is completed.

• virtual void setTFTFrameBuffer (uint16_t *address)=0

Sets the frame buffer address used by the TFT controller.

uint16_t * getClientFrameBuffer ()

Gets client frame buffer.

• virtual void touch (int32 t x, int32 t y)

Called by the touch driver to indicate a touch.

virtual void noTouch ()

Called by the touch driver to indicate that no touch is currently detected.

virtual void performDisplayOrientationChange ()

Perform the actual display orientation change.

Protected Attributes

• DMA_Interface & dma

A reference to the DMA interface.

· LCD & lcdRef

A reference to the LCD.

· TouchController & touchController

A reference to the touch controller.

• MCUInstrumentation * mcuInstrumentation

A reference to an optional MCU instrumentation.

• ButtonController * buttonController

A reference to an optional ButtonController.

• FrameBufferAllocator * frameBufferAllocator

A reference to an optional FrameBufferAllocator.

· Gestures gestures

Class for low-level interpretation of touch events.

DisplayOrientation nativeDisplayOrientation

Contains the native display orientation. If desired orientation is different, apply rotation.

void(* taskDelayFunc)(uint16_t)

Pointer to a function that can delay GUI task for a number of milliseconds.

uint16 t * frameBuffer0

Pointer to the first frame buffer.

• uint16_t * frameBuffer1

Pointer to the second frame buffer.

uint16_t * frameBuffer2

Pointer to the optional third frame buffer used for animation storage.

· FrameRefreshStrategy refreshStrategy

The selected display refresh strategy.

uint8_t fingerSize

The radius of the finger in pixels.

bool lockDMAToPorch

Whether or not to lock DMA transfers with TFT porch signal.

7.92 HAL Class Reference 379

· bool frameBufferUpdatedThisFrame

True if something was drawn in the current frame.

LCD * auxiliaryLCD

Auxiliary LCD class used to render Drawables into dynamic bitmaps.

Rect partialFrameBufferRect

The region of the screen covered by the partial framebuffer.

Static Protected Attributes

· static bool isDrawing

True if currently in the process of rendering a screen.

7.92.1 Detailed Description

Contains functions that are specific to the hardware platform the code is running on.

7.92.2 Member Enumeration Documentation

7.92.2.1 FrameRefreshStrategy

```
enum typedef enum FrameRefreshStrategy
```

See also

bool setFrameRefreshStrategy(FrameRefreshStrategy s)

Enumerator

REFRESH_STRATEGY_DEFAULT	If not explicitly set, this strategy is used.
REFRESH_STRATEGY_OPTIM_SINGLE_BUFFE↔	Strategy optimized for single frame buffer on systems
R_TFT_CTRL	with TFT controller.
REFRESH_STRATEGY_PARTIAL_FRAMEBUFFER	Strategy using less than a full frame buffer.

7.92.3 Constructor & Destructor Documentation

Creates a HAL instance.

Parameters

in	dmaInterface	Reference to the DMA interface.
in	display	Reference to the LCD.
in	touchCtrl	Reference to the touch controller.
	width	The width of the LCD display, in pixels.
	height	The height of the LCD display, in pixels.

```
7.92.3.2 ~HAL()
~HAL ( ) [inline], [virtual]
Destructor.
```

7.92.4 Member Function Documentation

```
7.92.4.1 allowDMATransfers()
void allowDMATransfers ( ) [virtual]
```

Allow the DMA to start transfers. Front Porch Entry is a good place to call this.

7.92.4.2 backPorchExited()

```
void backPorchExited ( ) [inline], [virtual]
```

Has to be called from within the LCD IRQ rutine when the Back Porch Exit is reached.

7.92.4.3 beginFrame()

```
bool beginFrame ( ) [protected], [virtual]
```

Called when beginning to rendering a frame.

Returns

true if rendering can begin, false otherwise.

7.92.4.4 blitCopy() [1/3]

7.92 HAL Class Reference 381

```
bool hasTransparentPixels,
uint16_t dstWidth,
Bitmap::BitmapFormat srcFormat,
Bitmap::BitmapFormat dstFormat )
```

Blits a 2D source-array to the frame buffer performing alpha-blending (and transparency keying) as specified.

Note

Alpha=255 is assumed "solid" and shall be used if HAL does not support BLIT_OP_COPY_WITH_ALPHA.

Parameters

pSrc	The source-array pointer (points to first value to copy)
pClut	The CLUT pointer (points to CLUT header data which include the type and size of this
	CLUT followed by colors data)
X	The destination x coordinate on the frame buffer.
у	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
srcWidth	The distance (in elements) from first value of first line, to first value of second line (the
	source 2D array width)
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.
dstWidth	The distance (in elements) from first value of first line, to first value of second line (the
	destination 2D array width)
srcFormat	The source buffer color format (default is the framebuffer format)
dstFormat	The destination buffer color format (default is the framebuffer format)

7.92.4.5 blitCopy() [2/3]

Blits a 2D source-array to the frame buffer performing alpha-blending (and transparency keying) as specified.

Note

Alpha=255 is assumed "solid" and shall be used if HAL does not support BLIT_OP_COPY_WITH_ALPHA.

pSrc	The source-array pointer (points to first value to copy)
------	--

Parameters

Х	The destination x coordinate on the frame buffer.
у	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
srcWidth	The distance (in elements) from first value of first line, to first value of second line (the source 2D array width)
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.
dstWidth	The distance (in elements) from first value of first line, to first value of second line (the destination 2D array width)
srcFormat	The source buffer color format (default is the framebuffer format)
dstFormat	The destination buffer color format (default is the framebuffer format)

7.92.4.6 blitCopy() [3/3]

Blits a 2D source-array to the frame buffer performing alpha-blending (and transparency keying) as specified using the default lcd format.

Note

Alpha=255 is assumed "solid" and shall be used if HAL does not support BLIT_OP_COPY_WITH_ALPHA.

pSrc	The source-array pointer (points to first value to copy)
Х	The destination x coordinate on the frame buffer.
у	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
srcWidth	The distance (in elements) from first value of first line, to first value of second line (the source 2D array width)
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.

7.92 HAL Class Reference 383

7.92.4.7 blitCopyARGB8888()

Blits a 2D source-array to the frame buffer performing per-pixel alpha blending.

Parameters

pSrc	The source-array pointer (points to first value to copy)
Χ	The destination x coordinate on the frame buffer.
У	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
srcWidth	The distance (in elements) from first value of first line, to first value of second line (the source 2D array width)
alpha	The alpha value to use for blending. This is applied on every pixel, in addition to the per-pixel alpha value (255 = solid, no blending)

7.92.4.8 blitCopyGlyph()

Blits a 4bpp or 8bpp glyph - maybe use the same method and supply additional color mode arg.

pSrc	The source-array pointer (points to first value to copy)
X	The destination x coordinate on the frame buffer.
У	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
srcWidth	The distance (in elements) from first value of first line, to first value of second line (the source 2D array width)
color	Color of the text.
alpha	The alpha value to use for blending (255 = solid, no blending)
operation	The operation type to use for blit copy.

Blits a color value to the frame buffer performing alpha-blending (and transparency keying) as specified.

Bitmap::BitmapFormat dstFormat) [virtual]

Note

Alpha=255 is assumed "solid" and shall be used if HAL does not support BLIT_OP_FILL_WITH_ALPHA.

Parameters

color	The desired fill-color.
X	The destination x coordinate on the frame buffer.
У	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.
height	The height of desired area of the source 2D array.
alpha	The alpha value to use for blending (255 = solid, no blending)
dstWidth	The distance (in elements) from first value of first line, to first value of second line (the destination 2D array width)
dstFormat	The destination buffer color format (default is the framebuffer format)

Blits a color value to the frame buffer performing alpha-blending (and transparency keying) as specified.

Note

Alpha=255 is assumed "solid" and shall be used if HAL does not support BLIT_OP_FILL_WITH_ALPHA.

color	The desired fill-color.
X	The destination x coordinate on the frame buffer.
У	The destination y coordinate on the frame buffer.
width	The width desired area of the source 2D array.

7.92 HAL Class Reference 385

Parameters

height	The height of desired area of the source 2D array.
alpha	The alpha value to use for blending (255 = solid, no blending)

7.92.4.11 blitSetTransparencyKey()

Only present for backwards compatibility in TouchGFX 4.x. Will be removed in TouchGFX 5.

Parameters

kev	The "transparent" color value.
	i i i i i i i i i i i i i i i i i i i

Reimplemented in HALSDL2.

7.92.4.12 blockCopy()

This function performs a platform-specific memcpy, if supported by the hardware.

Parameters

out	dest	Pointer to destination memory.
in	src	Pointer to source memory.
	numBytes	Number of bytes to copy.

Returns

true if the copy succeeded, false if copy was not performed.

Reimplemented in HALSDL2.

7.92.4.13 cacheTextString()

This function can be used to cache a given string in a platform specific way to e.g. speed up access or in case the string is placed in a memory type that does not support random access such as NAND flash.

str	A pointer to the string which may be in external memory.

Returns

A pointer to an identical string which is guaranteed to be directly readable (ie. a copy if the original string was placed in NAND flash).

7.92.4.14 configureInterrupts()

```
void configureInterrupts ( ) [pure virtual]
```

Configures the interrupts relevant for TouchGFX. This primarily entails setting the interrupt priorities for the DMA and LCD interrupts.

Implemented in HALSDL2.

7.92.4.15 configurePartialFrameBuffer()

Configures a partial framebuffer as current framebuffer. This method uses the assigned FrameBufferAllocator to allocate block of compatible dimensions. The height of the allocated block is returned.

Parameters

X	The absolute x coordinate of the block on the screen.
У	The absolute y coordinate of the block on the screen.
width	The width of the block.
height	The height of the block requested.

Returns

The height of the block allocated.

7.92.4.16 copyFBRegionToMemory() [1/2]

Copies a region of the currently displayed frame buffer to memory. Used for e.g. SlideTransition and for displaying pre-rendered drawables e.g. in animations where redrawing the drawable is not necessary.

Note

Requires animation storage to be present.

Returns

A pointer to the memory address containing the copy of the frame buffer.

7.92.4.17 copyFBRegionToMemory() [2/2]

Copies a region of the currently displayed frame buffer to a buffer. Used for e.g. SlideTransition and for displaying pre-rendered drawables e.g. in animations where redrawing the drawable is not necessary. The buffer can e.g. be a dynamic bitmap.

Note

Requires animation storage to be present.

Parameters

		The frame buffer region to copy.
		Address of the buffer to store the copy in.
Ī	stride	The width of the target buffer (row length).

Returns

A pointer to the memory address containing the copy of the frame buffer.

7.92.4.18 disableInterrupts()

```
void disableInterrupts ( ) [pure virtual]
```

Disables the DMA and LCD interrupts.

Implemented in HALSDL2.

7.92.4.19 drawDrawableInDynamicBitmap() [1/2]

Render a Drawable and its widgets into a dynamic bitmap.

Parameters

drawable	A reference on the Drawable object to render.
bitmapld	Dynamic bitmap to be used as a rendertarget.

7.92.4.20 drawDrawableInDynamicBitmap() [2/2]

Render a Drawable and its widgets into a dynamic bitmap. Only the specified Rect region is updated.

Parameters

drawable	A reference on the Drawable object to render.
bitmapld	Dynamic bitmap to be used as a rendertarget.
rect	Region to update.

7.92.4.21 enableInterrupts()

```
void enableInterrupts ( ) [pure virtual]
```

Enables the DMA and LCD interrupts.

Implemented in HALSDL2.

7.92.4.22 enableLCDControllerInterrupt()

```
void enableLCDControllerInterrupt ( ) [pure virtual]
```

Configure the LCD controller to fire interrupts at VSYNC. Called automatically once TouchGFX initialization has completed.

Implemented in HALSDL2.

7.92.4.23 enableMCULoadCalculation()

```
void enableMCULoadCalculation (
          bool enabled ) [inline]
```

This method sets a flag that determines if generic HAL should calculate MCU load based on concrete MCU instrumentation.

Parameters

enabled	If true, set flag to update MCU load.

7.92.4.24 endFrame()

```
void endFrame ( ) [protected], [virtual]
```

Called when a rendering pass is completed.

7.92.4.25 flushDMA()

```
void flushDMA ( ) [virtual]
```

This function blocks until the DMA queue (containing BlitOps) is empty.

```
7.92.4.26 flushFrameBuffer() [1/2]
```

```
void flushFrameBuffer ( ) [virtual]
```

On some platforms, a local frame buffer needs to be pushed to the display through a SPI channel or similar. Implement that functionality here. This function is called whenever the framework has performed a complete draw.

Reimplemented in HALSDL2.

7.92.4.27 flushFrameBuffer() [2/2]

This function is called whenever the framework has performed a partial draw.

Parameters

rect The area of the screen that has been drawn, expressed in absolute coordinates.

See also

flushFrameBuffer().

Reimplemented in HALSDL2.

7.92.4.28 frontPorchEntered()

```
void frontPorchEntered ( ) [inline]
```

Has to be called from within the LCD IRQ routine when the Front Porch Entry is reached.

7.92.4.29 getAnimationStorage()

```
uint16_t * getAnimationStorage ( ) const [inline]
```

Gets the optional frame buffer used for animation storage.

Returns

The address or 0 if unused.

7.92.4.30 getAuxiliaryLCD()

```
void getAuxiliaryLCD ( ) [inline]
```

Get the auxiliary LCD class attached to the HAL instance if any.

Returns

A pointer on the axiliary LCD class attached to the HAL instance.

7.92.4.31 getBlitCaps()

```
BlitOperations getBlitCaps ( ) [inline], [virtual]
```

Function for obtaining the blit capabilities of the concrete HAL implementation. As default, will return whatever blitcaps are reported by the associated DMA object.

Returns

a bitmask of the supported blitcaps.

7.92.4.32 getButtonController()

```
ButtonController * getButtonController ( ) const [inline]
```

Gets the associated ButtonController.

Returns

A pointer to the ButtonController, or zero if no ButtonController has been set.

7.92.4.33 getClientFrameBuffer()

```
uint16_t * getClientFrameBuffer ( ) [inline], [protected]
```

Gets client frame buffer.

Returns

The address of the framebuffer currently used by the framework to draw in.

7.92.4.34 getCPUCycles()

```
uint32_t getCPUCycles ( )
```

Gets the current cycle counter.

Returns

the cycle counter.

7.92.4.35 getDisplayHeight()

```
uint16_t getDisplayHeight ( ) const [inline]
```

Gets display height.

Returns

The display height.

7.92.4.36 getDisplayOrientation()

```
DisplayOrientation getDisplayOrientation ( ) const [inline]
```

Gets the current display orientation. Will be equal to the native orientation of the display unless setDisplayOrientation has been explicitly called earlier.

Returns

The current display orientation.

7.92.4.37 getDisplayWidth()

```
uint16_t getDisplayWidth ( ) const [inline]
```

Gets display width.

Returns

The display width.

7.92.4.38 getDMAType()

```
DMAType getDMAType ( ) [inline], [virtual]
```

Function for obtaining the DMA type of the concrete DMA implementation. As default, will return DMA_TYPE_ $G \leftarrow$ ENERIC type value.

Returns

a DMAType value of the concrete DMA implementation.

7.92.4.39 getFingerSize()

```
uint8_t getFingerSize ( ) const [inline]
```

Gets the finger size in pixels.

Returns

The size of the finger in pixels, 1 is the default value.

```
7.92.4.40 getFrameBufferAllocator()
```

```
FrameBufferAllocator * getFrameBufferAllocator ( ) [inline]
```

Gets the framebuffer allocator.

Returns

The framebuffer allocator

7.92.4.41 getFrameRefreshStrategy()

```
FrameRefreshStrategy getFrameRefreshStrategy ( ) const [inline]
```

Returns

Current frame refresh strategy.

See also

bool setFrameRefreshStrategy(FrameRefreshStrategy s)

```
7.92.4.42 getInstance()
```

```
static HAL * getInstance ( ) [inline], [static]
```

Gets the HAL instance.

Returns

The HAL instance.

7.92.4.43 getLCDRefreshCount()

```
uint32_t getLCDRefreshCount ( ) [inline]
```

Returns the number of VSync interrupts between the current drawing operation and the last drawing operation, i.e. the number of lost frames.

Returns

Number of VSync since previous draw.

```
7.92.4.44 getMCULoadPct()
```

```
uint8_t getMCULoadPct ( ) const [inline]
```

Gets the current MCU load.

Returns

mcuLoadPct the MCU Load in %.

7.92.4.45 getTFTCurrentLine()

```
uint16_t getTFTCurrentLine ( ) [inline], [virtual]
```

This function is used to obtain the progress of the TFT controller. More specifically, the line (or Y-value) currently being transferred.

Note: The value must be adjusted to account for vertical back porch before returning, such that the value is always within the range of $0 \le$ value \le actual display height in pixels

It is used for the REFRESH_STRATEGY_OPTIM_SINGLE_BUFFER_TFT_CTRL frame refresh strategy in order to synchronize frame buffer drawing with TFT controller progress. If this strategy is used, the concrete HAL subclass must provide an override of this function that returns correct line value. If this strategy is not used, then the getTF \leftarrow TCurrentLine function is never called and can be disregarded.

Returns

In this default implementation, 0xFFFF is returned to signify "not implemented".

7.92.4.46 getTFTFrameBuffer()

```
uint16_t * getTFTFrameBuffer ( ) const [pure virtual]
```

Gets the frame buffer address used by the TFT controller.

Returns

The address of the frame buffer currently being displayed on the TFT.

Implemented in HALSDL2.

7.92.4.47 getTouchSampleRate()

```
int8_t getTouchSampleRate ( ) const [inline]
```

Gets the number of ticks between each touch screen sample.

Returns

Number of ticks between each touch screen sample.

7.92.4.48 initialize()

```
void initialize ( )
```

This function is responsible for initializing the entire framework.

7.92.4.49 lcd()

```
static LCD & lcd ( ) [inline], [static]
```

Gets a reference to the LCD.

Returns

A reference to the LCD.

7.92.4.50 lockDMAToFrontPorch()

```
void lockDMAToFrontPorch (
                bool enableLock ) [inline]
```

Function to set whether the DMA transfers are locked to the TFT update cycle. If locked, DMA transfer will not begin until the TFT controller has finished updating the display. If not locked, DMA transfers will begin as soon as possible. Default is true (DMA is locked with TFT).

Disabling the lock will in most cases significantly increase rendering performance. It is therefore strongly recommended to disable it. Depending on platform this may in rare cases cause rendering problems (visible tearing on display). Please see the chapter "Optimizing DMA During TFT Controller Access" for details on this setting.

Note

This setting only has effect when using double buffering.

Parameters

enableLock	True to lock DMA transfers to the front porch signal. Conservative, default setting. False to
	disable, which will normally yield substantial performance improvement.

7.92.4.51 lockFrameBuffer()

```
uint16_t * lockFrameBuffer ( ) [virtual]
```

Waits for the frame buffer to become available for use (i.e. not used by DMA transfers).

Note

Function blocks until frame buffer is available. Client code MUST call unlockFrameBuffer() when frame buffer operation has completed.

Returns

A pointer to the beginning of the currently used frame buffer.

7.92.4.52 noTouch()

```
void noTouch ( ) [protected], [virtual]
```

Called by the touch driver to indicate that no touch is currently detected.

7.92.4.53 performDisplayOrientationChange()

```
void performDisplayOrientationChange ( ) [inline], [protected], [virtual]
```

Perform the actual display orientation change.

Reimplemented in HALSDL2.

7.92.4.54 registerEventListener()

Registers an event handler implementation with the underlying event system. The actual HAL implementation decides whether or not multiple UIEventListener instances are allowed (including execution order).

Parameters

in listener The listener to regis

7.92.4.55 registerTaskDelayFunction()

In order to make use of the HAL::taskDelay function, a delay function must be registered by calling this function. Usually the delay function would be OSWrappers::taskDelay.

Parameters

in,out	delayF	A pointer to a function returning void with an uint16_t parameter specifying number of	
		milliseconds to delay.	

Note

The task delay capability is only used when the frame refresh strategy REFRESH_STRATEGY_OPTIM_SI⊷ NGLE_BUFFER_TFT_CTRL is selected. Otherwise it is not necessary to register a delay function.

7.92.4.56 registerTextCache()

Configures HAL to use the supplied buffer as text string cache. The buffer must be large enough to hold the longest string in the system. Setting this buffer is only required if cacheTextString() is actually used and its implementation requires a buffer.

Parameters

in	str	Pointer to buffer location.
	length	Buffer length (in UnicodeChar's)

See also

cacheTextString

7.92.4.57 sampleKey()

Sample external key event.

Parameters

out	Output parameter that will be set to the key value if a keypress was	detected.
-----	--	-----------

Returns

True if a keypress was detected and the "key" parameter is set to a value.

Reimplemented in HALSDL2.

7.92.4.58 setAuxiliaryLCD()

Set an auxiliary LCD class to be used for offscreen rendering.

Parameters

auxLCD | A pointer on the axiliary LCD class to use for offscreen rendering.

7.92.4.59 setButtonController()

Stores a pointer to an instance of a specific implementation of a ButtonController.

Parameters

in	btnCtrl	pointer to button controller.
----	---------	-------------------------------

7.92.4.60 setDisplayOrientation()

Sets the desired display orientation (landscape or portrait). If desired orientation is different from the native orienta-

tion of the display, a rotation is automatically applied. The rotation does not incur any performance cost.

Note

A screen transition must occur before this takes effect!

Parameters

orientation	The desired display orientation.
-------------	----------------------------------

7.92.4.61 setDragThreshold()

Configure the threshold for reporting drag events. A touch input movement must exceed this value in either axis in order to report a drag. Default value is 0.

Note

Use if touch controller is not completely accurate to avoid "false" drags.

Parameters

value	New threshold value.
-------	----------------------

7.92.4.62 setFingerSize()

Sets the finger size in pixels.

Setting the finger size to a size of more than 1 pixel will emulate a finger of width and height of 2*(fingersize-1)+1. This can be especially useful when trying to interact with small elements on a high ppi display. The finger size will influence which element is chosen as the point of interaction, when clicking, dragging, ... the display. A number of samples will be drawn from within the finger area and a best matching drawable will be chosen. The best matching algorithm will consider the size of the drawable and the distance from the touch point.

Parameters

```
in size the size of the finger.
```

7.92.4.63 setFrameBufferAllocator()

Sets a framebuffer allocator. The framebuffer allocator is only used in partial framebuffer mode.

Parameters

in	allocator	pointer to a framebuffer allocator object
----	-----------	---

7.92.4.64 setFrameBufferStartAddress()

Sets the address used for frame buffers, usually located in external memory. Will reserve memory for one or two frame buffers based on display size. Will optionally also reserve memory for a third frame buffer used for animation

Storage.

Parameters

in adr		Starting address to use for frame buffers.	
	depth	(Optional) Depth of each pixel in bits, default is 16.	
	useDoubleBuffering	(Optional) If true, reserve memory for an extra frame buffer.	
	useAnimationStorage	(Optional) If true, reserve memory for animation storage.	

7.92.4.65 setFrameBufferStartAddresses()

Sets individual frame buffer start addresses.

Parameters

in	frameBuffer	Buffer for frame buffer data, must be non-null.	
in	doubleBuffer	If non-null, buffer for double buffer data. If null double buffering is disabled.	
in	animationStorage	If non-null, the animation storage. If null animation storage is disabled.	

7.92.4.66 setFrameRateCompensation()

```
void setFrameRateCompensation (
          bool enabled ) [inline]
```

Enables or disables compensation for lost frames. See knowledge base article.

Parameters

enabled	true to enable, false to disable.

7.92.4.67 setFrameRefreshStrategy()

By setting a different frame refresh strategy, the internals of how TouchGFX interacts with the frame buffer can be modified.

Currently there are two strategies available. This will increase over time.

- REFRESH_STRATEGY_OPTIM_SINGLE_BUFFER_TFT_CTRL: this strategy is available on targets that use single buffering on a TFT controller based system. It requires an implementation of the getTFTCurrent← Line() function as well as a task delay function being registered. The implementation of this strategy is that TouchGFX will carefully track the progress of the TFT controller, and draw parts of the frame buffer whenever possible. The effect is that the risk of tearing is much reduced compared to the default single buffer strategy of only drawing in porch areas. It does have a drawback of slightly increased MCU load. But in many cases employing this strategy will make it possible to avoid external RAM, by using just a single frame buffer in internal RAM and still avoid tearing.
- REFRESH_STRATEGY_DEFAULT: This is a general strategy that works for all target configurations.

Recommendation: Try using REFRESH_STRATEGY_OPTIM_SINGLE_BUFFER_TFT_CTRL if you're on a TFT controller based system (ie. non-8080) and you have a desire to avoid external RAM. Otherwise stick to REFRE← SH_STRATEGY_DEFAULT.

Parameters

s The desired strategy to use.

Returns

true if the desired strategy will be used, false otherwise.

7.92.4.68 setMCUActive()

Register if MCU is active by measuring cpu cycles. If user wishes to track MCU load, this method should be called whenever the OS Idle task is scheduled in or out. This method makes calls to a concrete implementation of GPIO functionality and a concrete implementation of cpu cycles.

Parameters

active If true, MCU is registered as being active, inactive otherwise.

7.92.4.69 setMCUInstrumentation()

Stores a pointer to an instance of an MCU specific instrumentation class.

Parameters

in	mcuInstr	pointer to MCU instrumentation.	
----	----------	---------------------------------	--

7.92.4.70 setTFTFrameBuffer()

Sets the frame buffer address used by the TFT controller.

Parameters

in	address	New frame buffer address.
----	---------	---------------------------

Implemented in HALSDL2.

7.92.4.71 setTouchSampleRate()

Sets the number of ticks between each touch screen sample.

Parameters

```
sampleRateInTicks Sample rate. Default is 1 (every tick).
```

7.92.4.72 signalDMAInterrupt()

```
void signalDMAInterrupt ( ) [inline]
```

Notify the framework that a DMA interrupt has occurred.

7.92.4.73 swapFrameBuffers()

```
void swapFrameBuffers ( )
```

Swaps the two frame buffers.

7.92.4.74 taskDelay()

This function requires the presence of a task delay function. If a task delay function has not been registered, it returns immediately. Otherwise it returns when number of milliseconds has passed.

Parameters

ms	Number of milliseconds to wait.
----	---------------------------------

See also

void registerTaskDelayFunction(void (*delayF)(uint16_t))

7.92.4.75 taskEntry()

```
void taskEntry ( ) [virtual]
```

Main event loop. Will wait for VSYNC signal, and then process next frame. Call this function from your GUI task.

Note

This function never returns!

Reimplemented in HALSDL2.

7.92.4.76 tick()

```
void tick ( ) [protected], [virtual]
```

This function is called at each timer tick, depending on platform implementation.

7.92.4.77 touch()

Called by the touch driver to indicate a touch.

Parameters

Χ	The x coordinate of the touch.
V	The y coordinate of the touch.

7.92.4.78 unlockFrameBuffer()

```
void unlockFrameBuffer ( ) [virtual]
```

Unlocks the frame buffer (MUST be called exactly once for each call to lockFrameBuffer()).

7.92.4.79 vSync()

```
void vSync ( ) [inline]
```

Called by the VSync interrupt for counting of LCD refreshes.

7.93 HALSDL2 Class Reference

HAL implementation for the TouchGFX simulator.

#include <platform/hal/simulator/sdl/HALSDL2.hpp>

Public Member Functions

HALSDL2 (DMA_Interface &dma, LCD &lcd, TouchController &touchCtrl, uint16_t width, uint16_t height)

Constructor. Initializes members.

virtual void taskEntry ()

Main event loop.

virtual bool sampleKey (uint8_t &key)

Sample key event from keyboard.

virtual void flushFrameBuffer ()

This function is called whenever the framework has performed a complete draw.

virtual void flushFrameBuffer (const Rect &rect)

This function is called whenever the framework has performed a partial draw.

virtual bool blockCopy (void *RESTRICT dest, const void *RESTRICT src, uint32_t numBytes)

This function performs a platform-specific memcpy.

virtual void blitSetTransparencyKey (uint16_t key)

If Blit-operations are supported, transparency-keying support is implicitly assumed.

void setVsyncInterval (float ms)

Sets vsync interval.

bool doSampleTouch (int32_t &x, int32_t &y) const

Samples the position of the mouse cursor.

• virtual bool sdl_init (int argcount, char **args)

Initializes SDL.

• void setWindowTitle (const char *title)

Sets window title.

• const char * getWindowTitle () const

Gets window title.

void loadSkin (touchgfx::DisplayOrientation orientation, int x, int y)

Loads a skin for a given display orientation.

· void saveScreenshot ()

Saves a screenshot.

virtual void saveNextScreenshots (int n)

Copy the next N screenshots to disk.

virtual void saveScreenshot (char *folder, char *filename)

Saves a screenshot.

virtual void copyScreenshotToClipboard ()

Copies the screenshot to clipboard.

Static Public Member Functions

static char ** getArgv (int *argc)

Gets the argc and argv.

• static uint8_t * scaleTo24bpp (uint8_t *dst, uint16_t *src, Bitmap::BitmapFormat format)

Scale framebuffer to 24bpp.

static uint8_t * doRotate (uint8_t *dst, uint8_t *src)

Rotates a framebuffer if the display is rotated.

Protected Member Functions

virtual uint16_t * getTFTFrameBuffer () const

Gets TFT frame buffer.

void setTFTFrameBuffer (uint16 t *addr)

Sets TFT frame buffer.

virtual void renderLCD_FrameBufferToMemory (const Rect &_rectToUpdate, uint8_t *frameBuffer)

Update frame buffer using an SDL Surface.

• virtual void disableInterrupts ()

Disables the DMA and LCD interrupts.

• virtual void enableInterrupts ()

Enables the DMA and LCD interrupts.

• virtual void configureLCDInterrupt ()

Configures LCD interrupt.

• virtual void enableLCDControllerInterrupt ()

Enables the LCD interrupt.

• virtual void configureInterrupts ()

Configures the interrupts relevant for TouchGFX.

• void performDisplayOrientationChange ()

Perform the actual display orientation change.

Additional Inherited Members

7.93.1 Detailed Description

HAL implementation for the TouchGFX simulator.

See also

HAL

7.93.2 Constructor & Destructor Documentation

7.93.2.1 HALSDL2()

```
HALSDL2 (

DMA_Interface & dma,

LCD & lcd,

TouchController & touchCtrl,

uint16_t width,

uint16_t height ) [inline]
```

Constructor. Initializes members.

Parameters

in	dma	Reference to DMA interface.
in	lcd	Reference to the LCD.
in	touchCtrl	Reference to Touch Controller driver.
	width	Width of the display.
	height	Height of the display.

7.93.3 Member Function Documentation

7.93.3.1 blitSetTransparencyKey()

If Blit-operations are supported, transparency-keying support is implicitly assumed.

Parameters

key	The "transparent" color value.
-----	--------------------------------

Reimplemented from HAL.

7.93.3.2 blockCopy()

This function performs a platform-specific memcpy, if supported by the hardware.

Parameters

out	dest	Pointer to destination memory.
	src	Pointer to source memory.
	numBytes	Number of bytes to copy.

Returns

true if the copy succeeded, false if copy was not performed.

Reimplemented from HAL.

7.93.3.3 configureInterrupts()

```
void configureInterrupts ( ) [inline], [protected], [virtual]
```

Configures the interrupts relevant for TouchGFX. This primarily entails setting the interrupt priorities for the DMA and LCD interrupts.

Implements HAL.

7.93.3.4 configureLCDInterrupt()

```
void configureLCDInterrupt ( ) [inline], [protected], [virtual]
```

Configures LCD interrupt.

7.93.3.5 copyScreenshotToClipboard()

```
void copyScreenshotToClipboard ( ) [virtual]
```

Copies the screenshot to clipboard.

7.93.3.6 disableInterrupts()

```
void disableInterrupts ( ) [inline], [protected], [virtual]
```

Disables the DMA and LCD interrupts.

Implements HAL.

7.93.3.7 doRotate()

Rotates a framebuffer if the display is rotated.

Parameters

out	dst	Destination for the rotated framebuffer. must be non-null if the screen is rotated.
in	src	The framebuffer.

Returns

Null if it fails, else a pointer to an uint8_t.

7.93.3.8 doSampleTouch()

Samples the position of the mouse cursor.

Parameters

out	X	The x coordinate.
out	у	The y coordinate.

Returns

True if touch detected, false otherwise.

7.93.3.9 enableInterrupts()

```
void enableInterrupts ( ) [inline], [protected], [virtual]
```

Enables the DMA and LCD interrupts.

Implements HAL.

7.93.3.10 enableLCDControllerInterrupt()

```
void enableLCDControllerInterrupt ( ) [inline], [protected], [virtual]
```

Enables the LCD interrupt.

Implements HAL.

```
7.93.3.11 flushFrameBuffer() [1/2]
```

```
void flushFrameBuffer ( ) [virtual]
```

On some platforms, a local frame buffer needs to be pushed to the display through a SPI channel or similar. Implement that functionality here. This function is called whenever the framework has performed a complete draw.

Reimplemented from HAL.

7.93.3.12 flushFrameBuffer() [2/2]

This function is called whenever the framework has performed a partial draw.

Parameters

rect The area of the screen that has been drawn, expressed in absolute coordinates.

See also

flushFrameBuffer(). This function is called whenever the framework has performed a partial draw.

Reimplemented from HAL.

7.93.3.13 getArgv()

Gets the argc and argv for a Windows program.

Parameters

argc Pointer to where to store number of arguments

Returns

The argv list of arguments.

7.93.3.14 getTFTFrameBuffer()

```
uint16_t * getTFTFrameBuffer ( ) const [protected], [virtual]
```

Gets TFT frame buffer.

Returns

null if it fails, else the TFT frame buffer.

Implements HAL.

7.93.3.15 getWindowTitle()

```
const char * getWindowTitle ( ) const
```

Gets window title.

Returns

null "TouchGFX simulator" unless set to something else using setWindowTitle().

See also

setWindowTitle

7.93.3.16 loadSkin()

Loads a skin for a given display orientation that will be rendered in the simulator window with the TouchGFX framebuffer placed inside the bitmap at the given coordinates. Different bitmaps can be loaded in landscape and portrait mode. If the provided bitmap cannot be loaded, the TouchGFX framebuffer will be displayed as normal. If the png files contain areas with alpha < 255, this will be used to create a shaped window.

Parameters

orientation	The orientation.
X	The x coordinate.
У	The y coordinate.

Note

The skins must be named "portrait.png" and "landscape.png" and placed inside the "simulator/" folder. The build process of the simulator will automatically copy the skins to the folder where the executable simulator is generated.

When as skin is set, the entire framebuffer is rendered through SDL whenever there is a change. Without a skin, only the areas with changes is rendered through SDL.

7.93.3.17 performDisplayOrientationChange()

```
void performDisplayOrientationChange ( ) [protected], [virtual]
```

Perform the actual display orientation change.

Reimplemented from HAL.

7.93.3.18 renderLCD_FrameBufferToMemory()

Update frame buffer using an SDL Surface.

Parameters

		_rectToUpdate	Area to update.
j	in	frameBuffer	Target frame buffer.

7.93.3.19 sampleKey()

Sample key event from keyboard.

Parameters

out	key	Output parameter that will be set to the key value if a key press was detected.
-----	-----	---

Returns

True if a key press was detected and the "key" parameter is set to a value.

Reimplemented from HAL.

7.93.3.20 saveNextScreenshots()

```
void saveNextScreenshots ( \quad \text{int } n \text{ ) } \quad [\text{virtual}]
```

Copy the next N screenshots to disk. On each screen update, the new screen is saved to disk.

Parameters

n Number of screenshots to save. These are added to any ongoing amount of screenshots in queue.

```
7.93.3.21 saveScreenshot() [1/2]
void saveScreenshot ( )
```

Saves a screenshot to the default folder and default filename.

Saves a screenshot.

Parameters

in	folder	Folder name to place the screenshot in.
in	filename	Filename to save the screenshot to.

7.93.3.23 scaleTo24bpp()

Scale framebuffer to 24bpp. The format of the framebuffer (src) is given in parameter format. The result is placed in the pre-allocated memory pointed to by parameter dst. If the frambebuffer is in format Bitmap::RGB888, parameter dst is not used and the parameter src is simply returned.

Parameters

out	dst	Destination for the framebuffer. must be non-null unless format is Bitmap::RGB888.
in	src	The framebuffer.
	format	Describes the format of the framebuffer (lcd().framebufferFormat()).

Returns

Null if it fails, else a pointer to an uint8_t.

7.93.3.24 sdl_init()

```
bool sdl_init (
```

```
int argcount,
char ** args ) [virtual]
```

Initializes SDL.

Parameters

	argcount	Number of arguments.
in	args	Arguments.

Returns

True if init went well, false otherwise.

7.93.3.25 setTFTFrameBuffer()

```
void setTFTFrameBuffer ( \label{eq:condition} \mbox{uint16\_t} \ * \ \mbox{addr} \ ) \ \ \mbox{[protected], [virtual]}
```

Sets TFT frame buffer.

Parameters

in	addr	The address of the TFT frame buffer.
----	------	--------------------------------------

Implements HAL.

7.93.3.26 setVsyncInterval()

Sets vsync interval for simulating same tick speed as the real hardware. Due to limitations in the granularity of SDL, the generated ticks in the simulator might not occur at the exact time, but accumulated over several ticks, the precision is very good.

Parameters

ms The milliseconds between	en ticks.
-----------------------------	-----------

Note

That you can also use HAL::setFrameRateCompensation() in the simulator. The effect of this can easily be demonstrated by dragging the console output window of the simulator (when running from Visual Studio) as this will pause the SDL and generate a lot of ticks when the console window is released. Beware that since the missed vsyncs are accumulated in an 8 bit counter, only up to 255 ticks may be missed, so at VsyncInterval = 16.6667, dragging the windows for more than 255 * 16.6667ms = 4250ms = 4.25s will not generate all the ticks that were actually missed. This situation is, however, not very realistic, as normally just a couple of vsyncs are skipped.

7.94 I2C Class Reference 411

7.93.3.27 setWindowTitle()

Sets window title of the TouchGFX simulator.

Parameters

```
title The title, if null the original "TouchGFX simulator" will be used.
```

See also

getWindowTitle

7.93.3.28 taskEntry()

```
void taskEntry ( ) [virtual]
```

Main event loop. Will wait for VSYNC signal, and then process next frame. Call this function from your GUI task.

Note

This function never returns!

Reimplemented from HAL.

7.94 I2C Class Reference

Platform independent interface for I2C drivers.

```
#include <platform/driver/i2c/I2C.hpp>
```

Public Member Functions

• I2C (uint8_t ch)

Stores the channel of the I2C bus to be configured.

virtual ∼I2C ()

Destructor.

• virtual void init ()=0

Initializes the I2C driver.

virtual bool readRegister (uint8_t addr, uint8_t reg, uint8_t *data, uint32_t cnt)=0

Reads the specified register on the device with the specified address.

• virtual bool writeRegister (uint8_t addr, uint8_t reg, uint8_t val)=0

Writes the specified value in a register.

Protected Attributes

• uint8_t channel

I2c channel is stored in order to initialize and recover a specific I2C channel.

7.94.1 Detailed Description

Platform independent interface for I2C drivers.

7.94.2 Constructor & Destructor Documentation

Stores the channel of the I2C bus to be configured.

Parameters

```
ch | I2C channel.
```

```
7.94.2.2 \simI2C() \simI2C ( ) [inline], [virtual] Destructor.
```

7.94.3 Member Function Documentation

```
7.94.3.1 init()
void init ( ) [pure virtual]
Initializes the I2C driver.
```

7.94.3.2 readRegister()

Reads the specified register on the device with the specified address.

Parameters

	addr	The I2C device address.
	reg	The register.
out	data	Pointer to buffer in which to place the result.
	cnt	Size of buffer in bytes.

Returns

true on success, false otherwise.

7.94.3.3 writeRegister()

```
bool writeRegister (
          uint8_t addr,
          uint8_t reg,
          uint8_t val ) [pure virtual]
```

Writes the specified value in a register.

Parameters

addr	The I2C device address.
reg	The register.
val	The new value.

Returns

true on success, false otherwise.

7.95 I2CTouchController Class Reference

Specific I2C-enabled type of Touch Controller.

```
#include <platform/driver/touch/I2CTouchController.hpp>
```

Public Member Functions

• I2CTouchController (I2C &i2c)

Constructor.

virtual ~I2CTouchController ()

Destructor.

• virtual void init ()=0

Initializes touch controller.

virtual bool sampleTouch (int32_t &x, int32_t &y)=0

Checks whether the touch screen is being touched.

Protected Attributes

I2C & i2c
 I2C driver.

7.95.1 Detailed Description

Specific I2C-enabled type of Touch Controller.

See also

TouchController

7.95.2 Constructor & Destructor Documentation

7.95.2.1 I2CTouchController()

Constructor. Initializes I2C driver.

Parameters

in,out <i>i2c</i> <mark>l2</mark>

7.95.2.2 ~I2CTouchController()

```
\simI2CTouchController ( ) [inline], [virtual]
```

Destructor.

7.95.3 Member Function Documentation

```
7.95.3.1 init()
```

```
void init ( ) [pure virtual]
```

Initializes touch controller.

Implements TouchController.

7.95.3.2 sampleTouch()

Checks whether the touch screen is being touched, and if so, what coordinates.

Parameters

out	Х	The x position of the touch
out	у	The y position of the touch

Returns

True if a touch has been detected, otherwise false.

Implements TouchController.

7.96 IconButtonStyle < T > Class Template Reference

An icon button style.

#include <touchgfx/containers/buttons/IconButtonStyle.hpp>

Public Member Functions

• IconButtonStyle ()

Default constructor.

virtual ∼IconButtonStyle ()

Destructor.

· virtual void setIconBitmaps (const Bitmap &newIconReleased, const Bitmap &newIconPressed)

Sets icon bitmaps.

void setIconX (int16_t x)

Sets icon x coordinate.

void setIconY (int16_t y)

Sets icon y coordinate.

void setIconXY (int16 t x, int16 t y)

Sets icon xy.

Bitmap getCurrentlyDisplayedIcon () const

Gets currently displayed icon.

int16_t getIconX () const

Gets icon x coordinate.

• int16_t getIconY () const

Gets icon y coordinate.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

· virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· Bitmap iconReleased

Icon to display when button is not pressed.

· Bitmap iconPressed

Icon to display when button is pressed.

· Image iconImage

The icon image.

7.96.1 Detailed Description

template < class T > class touchgfx::lconButtonStyle < T >

An icon button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show one of two icons depending on the state of the button (pressed or released).

To get a background behind the icon, use IconButtonStyle together with e.g. ImageButtonStyle: IconButton← Style<ImageButtonStyle<ClickButtonTrigger> > myButton;

The IconButtonStyle will center the icon on the enclosing container (normally AbstractButtonContainer). Set the size of the button before setting the icons.

The position of the icon can be adjusted with setIconXY.

Template Parameters

```
T Generic type parameter. Typically a AbstractButtonContainer subclass.
```

See also

AbstractButtonContainer

7.96.2 Member Function Documentation

```
7.96.2.1 getCurrentlyDisplayedIcon()
```

```
Bitmap getCurrentlyDisplayedIcon ( ) const [inline]
```

Returns

The currently displayed icon.

```
7.96.2.2 getlconX()
```

```
int16_t getIconX ( ) const [inline]
```

Returns

The icon x coordinate.

7.96.2.3 getlconY()

```
int16_t getIconY ( ) const [inline]
```

Returns

The icon y coordinate.

7.96.2.4 setIconBitmaps()

Parameters

newlconReleased	The new icon released.	
newlconPressed	The new icon pressed.	

7.96.2.5 setlconX()

```
void setIconX ( int16\_t \ x \ ) \quad [inline]
```

Parameters

7.96.2.6 setIconXY()

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.96.2.7 setIconY()

```
void setIconY (
          int16_t y ) [inline]
```

Parameters

```
y The y coordinate.
```

7.97 Image Class Reference

Simple widget capable of showing a bitmap.

```
#include <touchgfx/widgets/Image.hpp>
```

Public Member Functions

• Image (const Bitmap &bmp=Bitmap())

Default Constructor.

virtual void setBitmap (const Bitmap &bmp)

Sets the bitmap ID for this Image.

void setAlpha (uint8_t alpha)

Sets the alpha channel for the image.

· virtual void draw (const Rect &invalidatedArea) const

Draws the image.

• BitmapId getBitmap () const

Gets the Bitmapld currently contained by the widget.

uint8_t getAlpha () const

Gets the current alpha value.

• virtual Rect getSolidRect () const

Gets the largest solid (non-transparent) rectangle.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

Bitmap bitmap

The Bitmap to display.

uint8_t alpha

The Alpha for this image.

bool hasTransparentPixels

true if this object has transparent pixels

Additional Inherited Members

7.97.1 Detailed Description

Simple widget capable of showing a bitmap. The bitmap can be alpha-blended with the background and have areas of transparency.

See also

Widget

7.97.2 Constructor & Destructor Documentation

```
7.97.2.1 Image()

Image (
```

const Bitmap & bmp = Bitmap()) [inline]

Constructs a new Image with a default alpha value of 255 (solid) and a default Bitmap if none is specified.

Parameters

bmp The bitmap to display.

7.97.3 Member Function Documentation

7.97.3.1 draw()

Draws the image. This class supports partial drawing, so only the area described by the rectangle will be drawn.

Parameters

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.	
-----------------	--	--

Implements Drawable.

Reimplemented in TiledImage.

7.97.3.2 getAlpha()

```
uint8_t getAlpha ( ) const [inline]
```

Gets the current alpha value.

Returns

The current alpha value.

7.97.3.3 getBitmap()

```
BitmapId getBitmap ( ) const [inline]
```

Gets the BitmapId currently contained by the widget.

Returns

The current BitmapId of the widget.

7.97.3.4 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets the largest solid (non-transparent) rectangle. This value is pre-calculated by the imageconverter tool.

Returns

The largest solid (non-transparent) rectangle.

Implements Drawable.

Reimplemented in TiledImage.

7.97.3.5 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE IMAGE.

Reimplemented from Widget.

Reimplemented in AnimatedImage, and TiledImage.

7.97.3.6 setAlpha()

Sets the alpha channel for the image.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.97.3.7 setBitmap()

Sets the bitmap ID for this Image. Updates the width and height of this widget to match that of the bitmap.

Parameters

bmp The bitmap instan

See also

Bitmap

Reimplemented in AnimatedImage, and TiledImage.

7.98 ImageButtonStyle < T > Class Template Reference

An image button style.

#include <touchgfx/containers/buttons/ImageButtonStyle.hpp>

Public Member Functions

• ImageButtonStyle ()

Default constructor.

virtual ∼ImageButtonStyle ()

Destructor.

virtual void setBitmaps (const Bitmap &bmpReleased, const Bitmap &bmpPressed)

Sets the bitmaps.

void setBitmapXY (uint16_t x, uint16_t y)

Sets bitmap xy.

· Bitmap getCurrentlyDisplayedBitmap () const

Gets currently displayed bitmap.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

· virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· Image buttonImage

The button image.

· Bitmap up

The image to display when button is released.

· Bitmap down

The image to display when button is pressed.

7.98.1 Detailed Description

```
template < class T > class touchgfx::ImageButtonStyle < T >
```

An image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show one of two images depending on the state of the button (pressed or released).

The ImageButtonStyle will set the size of the enclosing container (normally AbstractButtonContainer) to the size of the pressed Bitmap. This can be overridden by calling setWidth/setHeight after setting the bitmaps.

The position of the bitmap can be adjusted with setBitmapXY (default is upper left corner).

Template Parameters

T Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer

7.98.2 Member Function Documentation

7.98.2.1 getCurrentlyDisplayedBitmap()

```
Bitmap getCurrentlyDisplayedBitmap ( ) const [inline]
```

Returns

The currently displayed bitmap.

7.98.2.2 setBitmaps()

Parameters

bmpReleased	The bitmap released.
bmpPressed	The bitmap pressed.

7.98.2.3 setBitmapXY()

Parameters

X	An uint16_t to process.
У	An uint16_t to process.

7.99 ImageProgress Class Reference

An image progress.

```
#include <ImageProgress.hpp>
```

Public Member Functions

• ImageProgress ()

Default constructor.

virtual ∼ImageProgress ()

Destructor.

• virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the position and dimension of the image progress indicator.

virtual void setAnchorAtZero (bool anchorAtZero)

Sets anchor at zero.

· virtual bool getAnchorAtZero () const

Gets anchor at zero.

virtual void setBitmap (touchgfx::BitmapId bitmapId)

Sets the bitmap id.

• virtual touchgfx::BitmapId getBitmap () const

Gets the image.

virtual void setAlpha (uint8_t alpha)

Sets the alpha.

• virtual uint8_t getAlpha () const

Gets the alpha.

virtual void setValue (int value)

Sets a value.

Protected Attributes

· Container container

The container.

· TiledImage image

The image.

· bool fixedPosition

true if the image should not move during progress

Additional Inherited Members

7.99.1 Detailed Description

touchgfx/containers/progress_indicators/ImageProgress.hpp

An image progress will show parts of an image as a progress indicator. The image can progress from the left, the right, the bottom or the top of the given area, and can visually be fixed with a larger and larger portion of the image showing, or it can be moved into view.

7.99.2 Constructor & Destructor Documentation

```
7.99.2.1 ImageProgress()
```

ImageProgress ()

Default constructor.

7.99.2.2 \sim ImageProgress()

 \sim ImageProgress () [virtual]

Destructor.

7.99.3 Member Function Documentation

```
7.99.3.1 getAlpha()
uint8_t getAlpha ( ) const [virtual]
Returns
     The the alpha of the image.
See also
     setAlpha
     Image::getAlpha
7.99.3.2 getAnchorAtZero()
bool getAnchorAtZero ( ) const [virtual]
Gets anchor at zero.
Returns
     true if the image is anchored at zero, false if it is anchored at current progress.
See also
     setAnchorAtZero
7.99.3.3 getBitmap()
touchgfx::BitmapId getBitmap ( ) const [virtual]
Gets the image.
Returns
     The image.
See also
     setBitmap
7.99.3.4 setAlpha()
void setAlpha (
              uint8_t alpha ) [virtual]
Sets the alpha of the image.
Parameters
```

alpha

The alpha.

See also

```
getAlpha
Image::setAlpha
```

7.99.3.5 setAnchorAtZero()

Sets anchor at zero will control whether the image will be placed so that it is not moving during progress, only more and more of the image will become visible, or if the image is anchored at the current progress and will appear to slide into view.

Parameters

```
anchorAtZero true to anchor at zero, false to anchor at current progress.
```

See also

getAnchoredAtZero

7.99.3.6 setBitmap()

Sets the bitmap id to use for progress. Please note that the bitmap is tiled which will allow smaller bitmaps to repeat and save memory.

Parameters

bitmap⇔	The bitmap id.
ld	

See also

getBitmap TiledImage

7.99.3.7 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the image progress indicator relative to the background image.

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the image progress indicator.
height	The height of the image progress indicator.

Reimplemented from AbstractProgressIndicator.

7.99.3.8 setValue()

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

value	The value.
-------	------------

Reimplemented from AbstractProgressIndicator.

7.100 InternalFlashFont Class Reference

An InternalFlashFont has both glyph table and glyph data placed in a flash which supports random access read (i.e. not a NAND flash).

```
#include <touchgfx/InternalFlashFont.hpp>
```

Public Member Functions

InternalFlashFont (const GlyphNode *list, uint16_t size, uint16_t height, uint8_t pixBelowBase, uint8_t bits
 PerPixel, uint8_t dataFormatA4, uint8_t maxLeft, uint8_t maxRight, const uint8_t *glyphDataInternalFlash, const KerningNode *kerningList, const Unicode::UnicodeChar fallbackChar, const Unicode::UnicodeChar ellipsisChar)

Constructor.

- virtual const uint8_t * getPixelData (const GlyphNode *glyph) const
 - Obtains a RAM-based pointer to the pixel data for the specified glyph.
- virtual int8_t getKerning (Unicode::UnicodeChar prevChar, const GlyphNode *glyph) const

Gets the kerning distance between two characters.

Additional Inherited Members

7.100.1 Detailed Description

An InternalFlashFont has both glyph table and glyph data placed in a flash which supports random access read (i.e. not a NAND flash)

See also

ConstFont

7.100.2 Constructor & Destructor Documentation

7.100.2.1 InternalFlashFont()

Construct the InternalFlashFont.

Parameters

list	The array of glyphs known to this font.
size	The number of glyphs in list.
height	The height in pixels of the highest character in this font.
pixBelowBase	The maximum number of pixels that can be drawn below the baseline in this font.
bitsPerPixel	The number of bits per pixel in this font.
dataFormatA4	The glyphs are saved using ST A4 format.
maxLeft	The maximum a character extends to the left.
maxRight	The maximum a character extends to the right.
glyphDataInternalFlash	Pointer to the glyph data for the font, placed in internal flash.
kerningList	pointer to the kerning data for the font, placed in internal flash.
fallbackChar	The fallback character for the typography in case no glyph is available.
ellipsisChar	The ellipsis character used for truncating long texts.

7.100.3 Member Function Documentation

7.100.3.1 getKerning()

Gets the kerning distance between two characters.

Parameters

prevChar	The unicode value of the previous character.
glyph	the glyph object for the current character.

Returns

The kerning distance between prevChar and glyph char.

Implements ConstFont.

7.100.3.2 getPixelData()

Obtains a RAM-based pointer to the pixel data for the specified glyph.

Parameters

glyph	The glyph to get the pixels data of.
-------	--------------------------------------

Returns

The pixel data of the glyph.

Implements ConstFont.

7.101 Scanline::iterator Class Reference

An iterator to help go through all the elements that make up a Scanline.

```
#include <touchgfx/canvas_widget_renderer/Scanline.hpp>
```

Public Member Functions

• iterator (const Scanline &scanline)

Constructor.

• int next ()

Gets the next element on the Scanline.

• int getNumPix () const

Gets number of consecutive pixels in the current run on the Scanline.

• const uint8_t * getCovers () const

Gets the covers in the current run on the Scanline.

7.101.1 Detailed Description

An iterator to help go through all the elements that make up a Scanline. Each part of the Scanline has a different Cover.

7.101.2 Constructor & Destructor Documentation

7.101.2.1 iterator()

Constructor. Creates an iterator to help go through all the Scanline parts of the polygon on a single Scanline.

Parameters

scanline The scanling	ne to iterate.
-----------------------	----------------

7.101.3 Member Function Documentation

7.101.3.1 getCovers()

```
const uint8_t * getCovers ( ) const [inline]
```

Gets the covers in the current run on the Scanline.

Returns

array of covers of each individual pixel.

7.101.3.2 getNumPix()

```
int getNumPix ( ) const [inline]
```

Gets number of consecutive pixels in the current run on the Scanline.

Returns

The number of consecutive pixels.

7.101.3.3 next()

```
int next ( ) [inline]
```

Gets the next element on the Scanline.

Returns

An the next index in the array of Scanline elements.

7.102 JSMOCHelper Class Reference

Helper class providing caching of certain information while the JSMOC algorithm runs during draw operations.

```
#include <touchgfx/JSMOCHelper.hpp>
```

Public Member Functions

• JSMOCHelper ()

Default constructor.

void setWidget (Drawable *newWidget)

Sets a widget.

Drawable * getWidget ()

Gets the widget.

• Rect & getCachedVisibleRect ()

Gets the visible rect for the widget of this helper.

• int16_t getCachedAbsX ()

Gets the absolute x coordinate for the widget of this helper.

• int16_t getCachedAbsY ()

Gets the absolute y coordinate for the widget of this helper.

• int16_t getWidth ()

Gets the width of the widget of this helper.

• int16_t getHeight ()

Gets the height of the widget of this helper.

void draw (const Rect &invalidatedArea)

Draws the widget of this helper.

7.102.1 Detailed Description

Helper class providing caching of certain information while the JSMOC algorithm runs during draw operations. Not intented for application-level use.

7.102.2 Constructor & Destructor Documentation

```
7.102.2.1 JSMOCHelper()
```

```
JSMOCHelper ( ) [inline]
```

Default constructor.

7.102.3 Member Function Documentation

```
7.102.3.1 draw()
```

Draws the widget of this helper.

Parameters

<i>invalidatedArea</i> ∣ The area of the widget to draw.	invalidatedArea	The area of the widget to draw.
--	-----------------	---------------------------------

7.102.3.2 getCachedAbsX()

```
int16_t getCachedAbsX ( ) [inline]
```

Gets the absolute x coordinate for the widget of this helper.

Returns

The absolute x coordinate for the widget of this helper. Only calculated once.

7.102.3.3 getCachedAbsY()

```
int16_t getCachedAbsY ( ) [inline]
```

Gets the absolute y coordinate for the widget of this helper.

Returns

The absolute y coordinate for the widget of this helper. Only calculated once.

7.102.3.4 getCachedVisibleRect()

```
Rect & getCachedVisibleRect ( ) [inline]
```

Gets the visible rect for the widget of this helper.

Returns

The visible rect for the widget of this helper. Only calculated once.

7.102.3.5 getHeight()

```
int16_t getHeight ( ) [inline]
```

Gets the height of the widget of this helper.

Returns

The height of the widget of this helper.

7.102.3.6 getWidget()

```
Drawable * getWidget ( ) [inline]
```

Gets the widget.

Returns

The widget this helper operates on.

7.102.3.7 getWidth()

```
int16_t getWidth ( ) [inline]
```

Gets the width of the widget of this helper.

Returns

The width of the widget of this helper.

7.102.3.8 setWidget()

Sets a widget.

Parameters

in	newWidget	The widget to operate on.
----	-----------	---------------------------

7.103 KerningNode Struct Reference

Structure providing information about a kerning for a given char pair.

```
#include <touchgfx/Font.hpp>
```

Public Attributes

• Unicode::UnicodeChar unicodePrevChar

The unicode for the first character in the kerning pair.

• int8_t distance

The kerning distance.

7.103.1 Detailed Description

Structure providing information about a kerning for a given char pair. Used by LCD when rendering.

7.104 Keyboard::Key Struct Reference

Mapping from rectangle to key id.

```
#include <touchgfx/widgets/Keyboard.hpp>
```

Public Attributes

uint8_t keyld

The id of a key.

· Rect keyArea

The area occupied by the key.

· Bitmapld highlightBitmapld

A bitmap to show when the area is "pressed".

7.105 Keyboard Class Reference

The keyboard provides text input for touch devices.

#include <touchgfx/widgets/Keyboard.hpp>

Classes

struct CallbackArea

Mapping from rectangle to a callback method to execute.

struct Key

Mapping from rectangle to key id.

struct KeyMapping

Mapping from key id to Unicode character.

struct KeyMappingList

List of KeyMappings to use.

· struct Layout

Definition of the keyboard layout. The keyboard can handle changing layouts, so different keyboard modes can be implemented by changing layouts and key mappings.

Public Member Functions

• Keyboard ()

Default Constructor.

virtual ∼Keyboard ()

Destructor.

• void setBuffer (Unicode::UnicodeChar *newBuffer, uint16 t newBufferSize)

Sets the buffer to be used by the keyboard.

void setLayout (const Layout *newLayout)

Set/change the Keyboard::Layout to use.

void setTextIndentation ()

Sets text indentation.

const Layout * getLayout () const

Gets the layout.

void setKeymappingList (const KeyMappingList *newKeyMappingList)

Set/change the KeyMappingList to use.

const KeyMappingList * getKeyMappingList () const

Gets key mapping list.

void setBufferPosition (uint16 t newPos)

Change the buffer position.

uint16 t getBufferPosition ()

Gets buffer position.

Unicode::UnicodeChar * getBuffer () const

Gets the buffer.

· virtual void draw (const Rect &invalidatedArea) const

Overrides the draw implementation on the Container.

virtual void handleClickEvent (const ClickEvent &evt)

Overrides the handleClickEvent on the container.

virtual void handleDragEvent (const DragEvent &evt)

Overrides the handleDragEvent on the container.

void setKeyListener (GenericCallback
 Unicode::UnicodeChar > &callback)

Sets the callback for the keyboard.

• virtual uint16_t getType () const

For GUI testing only.

Protected Member Functions

Key getKeyForCoordinates (int16_t x, int16_t y) const

Gets key for coordinates.

Unicode::UnicodeChar getCharForKey (uint8 t keyld) const

Maps a keyld to the UnicodeChar being displayed by that key.

CallbackArea getCallbackAreaForCoordinates (int16_t x, int16_t y) const

Gets the callback area defined by the layout for the specified coordinates.

virtual void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement)

Add to draw chain.

Protected Attributes

• GenericCallback< Unicode::UnicodeChar > * keyListener

Pointer to callback being executed when a key is pressed.

Unicode::UnicodeChar * buffer

Pointer to zero-terminated buffer where the entered text is being displayed.

uint16_t bufferSize

Size of the buffer.

uint16_t bufferPosition

Current position in buffer.

· Image image

Layout bitmap.

· TextAreaWithOneWildcard enteredText

Widget capable of displaying the entered text buffer.

· const Layout * layout

Pointer to layout.

const KeyMappingList * keyMappingList

Pointer to key mapping.

· Image highlightImage

Image to display when a key is highlighted.

bool cancellsEmitted

Tells if a cancel is emitted to check when a key is released.

Additional Inherited Members

7.105.1 Detailed Description

The keyboard provides text input for touch devices. It is configured using a Layout and a KeyMappingList, which both can be changed at runtime. The class using the keyboard must provide a buffer where the entered text is placed. The Layout contains a bitmap id for the image to display and two mappings: rectangles to key ids and rectangles to callback methods.

The KeyMappingList maps key ids to unicode characters. When the user presses a key, the keyboard looks in its layout for a rectangle containing the coordinates pressed. If it finds a mapping to a callback method, it will invoke that method. If it finds a mapping to a key it will look up the unicode character for that key and place it in a text buffer. The sequence is: (x,y) -> KeyId -> UnicodeChar.

A keyboard with multiple key mappings e.g. lower case alpha, upper case alpha and numeric mappings can be created by implementing callback methods for shift and mode areas in the provided bitmap and then changing the KeyMappingList when those areas are pressed.

See also

Container

7.105.2 Constructor & Destructor Documentation

```
7.105.2.1 Keyboard()
Keyboard ( )
Creates a new Keyboard.
7.105.2.2 ~Keyboard()
~Keyboard ( ) [inline], [virtual]
Destructor.
```

7.105.3 Member Function Documentation

Overrides the draw implementation on the Container. First invokes the container draw implementation to draw the keyboard bitmap and text area holding the entered text. If additional drawables have been added to the keyboard, they will also be draw. After invoking the container draw, the glyphs mapped to keys are drawn and if a key has been pressed, it will be highlighted.

Parameters

invalidatedArea T	ne area to draw.
-------------------	------------------

Reimplemented from Container.

```
7.105.3.2 getBuffer()
Unicode::UnicodeChar * getBuffer ( ) const [inline]
Gets the buffer.
```

Returns

The buffer containing entered text currently being displayed.

7.105.3.3 getBufferPosition()

```
uint16_t getBufferPosition ( ) [inline]
```

Gets buffer position.

Returns

the buffer position, i.e. the current index where new characters will be placed.

7.105.3.4 getCallbackAreaForCoordinates()

```
CallbackArea getCallbackAreaForCoordinates (
    int16_t x,
    int16_t y ) const [protected]
```

Gets the callback area defined by the layout for the specified coordinates.

Parameters

X	The x coordinate to perform key look up with.
У	The y coordinate to perform key look up with.

Returns

The CallbackArea, which is empty if not found.

7.105.3.5 getCharForKey()

Maps a keyld to the UnicodeChar being displayed by that key.

Parameters

key⇔	The id of the key to perform lookup with.
ld	

Returns

the UnicodeChar used for the specified key.

7.105.3.6 getKeyForCoordinates()

Gets key for coordinates.

Parameters

	The x coordinate to perform key look up with.
У	The y coordinate to perform key look up with.

Returns

The key for the given coordinates.

7.105.3.7 getKeyMappingList()

```
const KeyMappingList * getKeyMappingList ( ) const [inline]
```

Gets key mapping list.

Returns

The KeyMappingList used by the Keyboard.

7.105.3.8 getLayout()

```
const Layout * getLayout ( ) const [inline]
```

Gets the layout.

Returns

The layout used by the Keyboard.

7.105.3.9 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_KEYBOARD.

Reimplemented from Container.

7.105.3.10 handleClickEvent()

Overrides the handleClickEvent on the container. The keyboard handles all click events internally and click events are *not* propagated to drawables added to the keyboard.

Parameters

```
evt The ClickEvent.
```

Reimplemented from Drawable.

7.105.3.11 handleDragEvent()

Overrides the handleDragEvent on the container. The keyboard handles drag events to enable the container to, emit a CANCEL, if the user drags outside the currently pressed key.

Parameters

evt	The DragEvent.
-----	----------------

Reimplemented from Drawable.

7.105.3.12 setBuffer()

Sets the buffer to be used by the keyboard.

Parameters

in	newBuffer	Pointer to a buffer holding the text edited by the keyboard. If the buffer is not empty, the edit position for the keyboard will be set to the end of the provided text.
	newBufferSize	Length of the buffer, i.e. number of UnicodeChar's.

7.105.3.13 setBufferPosition()

Change the buffer position i.e. the next index to place a character when a key is pressed. This can be used to implement backspace functionality if the class using the Keyboard implements a callback and maps it to a backspace implementation. Setting the position will cause the TextArea displaying the text to be invalidated to request a redraw.

Parameters

newPos	The buffer position.
--------	----------------------

7.105.3.14 setKeyListener()

Sets the callback for the keyboard. The callback will be executed every time a key is clicked. The callback argument contains the key that was just pressed.

Note

Backspace, shift and mode keys report a 0 as value.

Parameters

in	callback	The Callback to invoke.
----	----------	-------------------------

7.105.3.15 setKeymappingList()

Set/change the KeyMappingList to use. The Keyboard will invalidate the space it occupies to request a redraw.

Parameters

7.105.3.16 setLayout()

Set/change the Keyboard::Layout to use. The Keyboard will invalidate the space it occupies to request a redraw.

Parameters

```
newLayout The new layout.
```

7.105.3.17 setTextIndentation()

```
void setTextIndentation ( )
```

Sets text indentation by making the area for entered text slightly larger. The result is that some characters (often

'j' and '_') will not be cut off. Indentation is added to both sides of the text area in case the text is right-to- left. Indentation is automatically set so all characters will display properly.

7.105.3.18 setupDrawChain()

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

Reimplemented from Container.

7.106 Keyboard::KeyMapping Struct Reference

Mapping from key id to Unicode character.

```
#include <touchgfx/widgets/Keyboard.hpp>
```

Public Attributes

· uint8_t keyld

Id of a key.

• Unicode::UnicodeChar keyValue

Unicode equivalent of the key id.

7.107 Keyboard::KeyMappingList Struct Reference

List of KeyMappings to use.

```
#include <touchgfx/widgets/Keyboard.hpp>
```

Public Attributes

const KeyMapping * keyMappingArray

The array of key mappings used by the keyboard.

uint8_t numberOfKeys

The number of keys in the list.

7.108 Keyboard::Layout Struct Reference

Definition of the keyboard layout. The keyboard can handle changing layouts, so different keyboard modes can be implemented by changing layouts and key mappings.

#include <touchgfx/widgets/Keyboard.hpp>

Public Attributes

· Bitmapld bitmap

The bitmap used for the keyboard layout.

const Key * keyArray

The keys on the keyboard layout.

• uint8_t numberOfKeys

The number of keys on this keyboard layout.

CallbackArea * callbackAreaArray

The array of areas and corresponding callbacks.

• uint8_t numberOfCallbackAreas

The number of areas and corresponding callbacks.

· Rect textAreaPosition

The area where text is written.

TypedText textAreaFont

The font used for typing text.

• colortype textAreaFontColor

The color used for the typing text.

FontId keyFont

The font used for the keys.

· colortype keyFontColor

The color used for the keys.

7.109 LCD Class Reference

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles.

#include <touchgfx/lcd/LCD.hpp>

Classes

struct StringVisuals

The visual elements when writing a string.

Public Member Functions

virtual ~LCD ()

Destructor.

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)=0

Draws a portion of a bitmap.

• virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)=0

Blits a 2D source-array to the framebuffer.

• virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)=0

Blits a 2D source-array to the frame buffer while converting the format.

uint16_t * copyFrameBufferRegionToMemory (const Rect ®ion, const BitmapId bitmapId=BITMAP_ANI

MATION_STORAGE)

Copies a part of the frame buffer.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)=0

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)=0

Draws a filled rectangle in the specified color.

void drawHorizontalLine (int16_t x, int16_t y, uint16_t width, uint16_t lineWidth, colortype color, uint8_

 t alpha=255)

Draws a horizontal line with the specified color.

 void drawVerticalLine (int16_t x, int16_t y, uint16_t height, uint16_t lineWidth, colortype color, uint8_t alpha=255)

Draws a vertical line with the specified color.

void drawRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a rectangle using the specified line color.

void drawBorder (const Rect &rect, uint16_t lineWidth, colortype color, uint8_t alpha=255)

Draws a rectangle width the specified line width and color.

 void drawString (Rect widgetArea, const Rect &invalidatedArea, const StringVisuals &stringVisuals, const Unicode::UnicodeChar *format,...)

Draws the specified unicode string.

virtual uint8_t bitDepth () const =0

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const =0

Framebuffer format used by the display.

virtual uint16_t framebufferStride () const =0

Framebuffer stride in bytes.

virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const =0

Generates a color representation to be used on the LCD, based on 24 bit RGB values. Depending on your chosen color bit depth, the color will be interpreted internally as either a 16 bit or 24 bit color value.

• virtual uint8_t getRedColor (colortype color) const =0

Gets the red color part of a color.

virtual uint8 t getGreenColor (colortype color) const =0

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const =0

Gets the blue color part of a color.

void drawTextureMapTriangle (const DrawingSurface &dest, const Point3D *vertices, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8

_t alpha=255, uint16_t subDivisionSize=12)

Texture map triangle. Draw a perspective correct texture mapped triangle. The vertices describes the surface, the x,y,z coordinates and the u,v coordinates of the texture. The texture contains the image data to be drawn The triangle line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the triangle should be alpha blended. The subDivisionSize will determine the size of the piecewise affine texture mapped portions of the triangle.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint8_t div255 (uint16_t num)

Approximates an integer division of a 16bit value by 255.

• static FORCE INLINE FUNCTION uint32 t div255rb (uint32 t pixelxAlpha)

Divides the red and blue components of pixelxAlpha by 255.

static FORCE INLINE FUNCTION uint32 t div255g (uint32 t pixelxAlpha)

Divides the green component of pixelxAlpha by 255.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionLength)=0

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_

t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data

FormatA4, colortype color, uint8 t bitsPerPixel, uint8 t alpha, TextRotation rotation)=0

Private version of draw-glyph.

 void drawStringLTR (const Rect &widgetArea, const Rect &invalidatedArea, const StringVisuals &visuals, const Unicode::UnicodeChar *format, va list pArg)

Draws the specified unicode string.

 void drawStringRTL (const Rect &widgetArea, const Rect &invalidatedArea, const StringVisuals &visuals, const Unicode::UnicodeChar *format, va list pArg)

Draws the specified unicode string.

Static Protected Member Functions

• static void rotateRect (Rect &rect, const Rect &canvas, const TextRotation rotation)

Rotate a rectangle inside another rectangle.

static int realX (const Rect &widgetArea, int16_t x, int16_t y, TextRotation rotation)

Find the real, absolute x coordinate of a point inside a widget.

• static int realY (const Rect &widgetArea, int16_t x, int16_t y, TextRotation rotation)

Find the real, absolute y coordinate of a point inside a widget.

static uint16_t stringWidth (TextProvider &textProvider, const Font &font, const int numChars, TextDirection textDirection)

Find string width.

Gets number lines.

Static Protected Attributes

• static const uint16_t newLine = 10

NewLine value.

7.109.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles.

Note

All coordinates are expected to be in absolute coordinates!

7.109.2 Constructor & Destructor Documentation

```
7.109.2.1 ~LCD()
~LCD ( ) [inline], [virtual]
Destructor.
```

7.109.3 Member Function Documentation

```
7.109.3.1 bitDepth()
uint8_t bitDepth ( ) const [pure virtual]
```

Returns

The number of bits per pixel.

Number of bits per pixel used by the display.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must
	be stored in a format suitable for the selected display.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD4bpp, LCD8bpp_AB⇔ GR2222, LCD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, and LCD1bpp.

```
const Rect & source,
const Rect & blitRect,
uint8_t alpha,
bool hasTransparentPixels ) [pure virtual]
```

Blits a 2D source-array to the frame buffer performing alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD4bpp, LCD8bpp_AB⇔ GR2222, LCD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, and LCD1bpp.

7.109.3.4 copyFrameBufferRegionToMemory() [1/2]

Copies a part of the frame buffer to a bitmap. The bitmap must be a dynamic bitmap or animation storage (default). Only the part specified with by parameter region is copied.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

regio	The part to copy.
bitma	OCO (Optional) The bitmap to store the data in. Default parameter is Animation Storage.
ld	

Returns

A pointer to the copy.

See also

blitCopy

7.109.3.5 copyFrameBufferRegionToMemory() [2/2]

```
\verb|uint16_t| * \verb|copyFrameBufferRegionToMemory| (
```

```
const Rect & visRegion,
const Rect & absRegion,
const BitmapId bitmapId ) [pure virtual]
```

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.6 div255()

Divides numerator num (e.g. the sum resulting from an alpha-blending operation) by 255.

Parameters

in	num	The numerator to divide by 255.

Returns

The result of a division by 255.

7.109.3.7 div255g()

Divides the green component of pixelxAlpha by 255.

Parameters

|--|

Returns

pixelxAlpha with its green component divided by 255.

7.109.3.8 div255rb()

Divides the red and blue components of pixelxAlpha by 255.

Parameters

in	pixelxAlpha	The red and blue components of a 32bit ARGB pixel multiplied by an alpha factor.
----	-------------	--

Returns

pixelxAlpha with its red and blue components divided by 255.

7.109.3.9 drawBorder()

Draws a rectangle width the specified line width and color.

Parameters

rect	The rectangle x, y, width, height in absolute coordinates.
lineWidth	The width of the line.
color	The color to use.
alpha	The alpha value to use (default=solid)

7.109.3.10 drawGlyph()

```
const Rect & invalidatedArea,
const GlyphNode * glyph,
const uint8_t * glyphData,
uint8_t dataFormatA4,
colortype color,
uint8_t bitsPerPixel,
uint8_t alpha,
TextRotation rotation ) [protected], [pure virtual]
```

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

out	wbuf	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw inside.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD32bpp, LCD24bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.11 drawHorizontalLine()

```
void drawHorizontalLine (
    int16_t x,
    int16_t y,
    uint16_t width,
    uint16_t lineWidth,
    colortype color,
    uint8_t alpha = 255 )
```

Draws a horizontal line with the specified color.

Parameters

X	The x coordinate of the starting point, in absolute coordinates.
У	The y coordinate of the starting point, in absolute coordinates.
width	The length of the line.
lineWidth	The width of the line.
color	The color to use.
alpha	The alpha value to use (default=solid)

7.109.3.12 drawPartialBitmap()

Draws a portion of a bitmap.

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0,0) on the screen.
У	The absolute y coordinate to place pixel (0,0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD4bpp, LCD8bpp_AB⇔ GR2222, LCD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, and LCD1bpp.

7.109.3.13 drawRect()

Draws a rectangle using the specified line color. Same as calling drawBorder with a line width of 1.

Parameters

rect	The rectangle x, y, width, height in absolute coordinates.
color	The color to use.
alpha	The alpha value to use (default=solid)

7.109.3.14 drawString()

```
void drawString (
    Rect widgetArea,
    const Rect & invalidatedArea,
    const StringVisuals & stringVisuals,
    const Unicode::UnicodeChar * format,
```

Draws the specified unicode string. Breaks line on newline.

Parameters

widgetArea	The area covered by the drawing widget in absolute coordinates.
invalidatedArea	The (sub)region of the widget area to draw, expressed relative to the widget area. If the widgetArea is (x, y, width, height) = (10, 10, 20, 20) and invalidatedArea is (x, y, width, height) = (5, 5, 6, 6) the widgetArea drawn on the LCD is (x, y, width, height) = (15, 15, 6, 6).
stringVisuals	The string visuals (font, alignment, line space, color) with which to draw this string.
format	A pointer to a zero terminated text string with optional additional wildcard arguments.
	Variable arguments providing additional information.

7.109.3.15 drawStringLTR()

Draws the specified unicode string. Breaks line on newline. The string is assumed to contain only latin characters written left-to-right.

Parameters

	widgetArea	The area covered by the drawing widget in absolute coordinates.
	invalidatedArea	The (sub)region of the widget area to draw, expressed relative to the widget area. If the widgetArea is (x, y, width, height) = (10, 10, 20, 20) and invalidatedArea is (x, y, width, height) = (5, 5, 6, 6) the widgetArea drawn on the LCD is (x, y, width, height) = (15, 15, 6, 6).
in	visuals	The string visuals (font, alignment, line space, color) with which to draw this string.
	format	A pointer to a zero terminated text string with optional additional wildcard arguments.
	pArg	Variable arguments providing additional information.

See also

drawString

7.109.3.16 drawStringRTL()

Draws the specified unicode string. Breaks line on newline. The string can be either right-to-left or left-to-right and may contain sequences of Arabic /Hebrew and Latin characters.

Parameters

	widgetArea	The area covered by the drawing widget in absolute coordinates.
--	------------	---

Parameters

	invalidatedArea	The (sub)region of the widget area to draw, expressed relative to the widget area. If the widgetArea is (x, y, width, height) = (10, 10, 20, 20) and invalidatedArea is (x, y, width, height) = (5, 5, 6, 6) the widgetArea drawn on the LCD is (x, y, width, height) = (15, 15, 6, 6).
in	visuals	The string visuals (font, alignment, line space, color) with which to draw this string.
	format	A pointer to a zero terminated text string with optional additional wildcard arguments.
	pArg	Variable arguments providing additional information.

See also

drawString

7.109.3.17 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionLength	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD24bpp, LCD32bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.18 drawTextureMapTriangle()

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
vertices	The vertices of the triangle.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	the alpha. Default = 255.
subDivisionSize	the size of the subdivisions of the scan line. Default = 12.

7.109.3.19 drawVerticalLine()

```
void drawVerticalLine (
    int16_t x,
    int16_t y,
    uint16_t height,
    uint16_t lineWidth,
    colortype color,
    uint8_t alpha = 255 )
```

Draws a vertical line with the specified color.

Parameters

X	The x coordinate of the starting point, in absolute coordinates.
У	The y coordinate of the starting point, in absolute coordinates.
height	The length of the line.
lineWidth	The width of the line.
color	The color to use.
alpha	The alpha value to use (default=solid)

7.109.3.20 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.	
color	The rectangle color. The rectangle opacity (255=solid)	
alpha		

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.21 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [pure virtual]
```

Framebuffer format used by the display

Returns

A Bitmap::BitmapFormat.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L ← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.22 framebufferStride()

```
uint16_t framebufferStride ( ) const [pure virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L ← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.23 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The 16 bit color value.

Returns

The blue part of the color.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD32bpp, LCD24bpp, LCD8bpp_ABGR2222, L← CD8bpp ARGB2222, LCD8bpp BGRA2222, LCD8bpp RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.24 getColorFrom24BitRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values. Depending on your chosen color bit depth, the color will be interpreted internally as either a 16 bit or 24 bit color value. This function can be safely used regardless of whether your application is configured for 16 or 24 bit colors.

Parameters

red	Value of the red part (0-255).	
green	Value of the green part (0-255).	
blue	Value of the blue part (0-255).	

Returns

The color representation depending on LCD color format.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD24bpp, LCD32bpp, LCD2bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.25 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD32bpp, LCD24bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109 LCD Class Reference 455

7.109.3.26 getNumLines()

Gets number of lines for a given text taking word wrap into consideration. The font and width are required to find the number of lines in case word wrap is true.

Parameters

in	textProvider	The text provider.	
wide TextAction The wide text action in case lines are longer than the width of		The wide text action in case lines are longer than the width of the text area.	
textDirection The text direction (LTR or RTL).		The text direction (LTR or RTL).	
font The font. width The width.		The font.	
		The width.	

Returns

The number lines.

7.109.3.27 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

1 -		ne color	
\perp core	<i>r</i> ⊢ ir	ne color	value
00,0		10 00101	valuo.

Returns

The red part of the color.

Implemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD32bpp, LCD24bpp, LCD8bpp_ABGR2222, L← CD8bpp_ARGB2222, LCD8bpp_BGRA2222, LCD8bpp_RGBA2222, LCD4bpp, and LCD1bpp.

7.109.3.28 init()

```
void init ( ) [inline], [virtual]
```

Performs initialization.

Reimplemented in LCD16bppSerialFlash, LCD16bpp, LCD2bpp, LCD2bpp, LCD24bpp, LCD32bpp, LCD8bpp_A⇔ BGR2222, LCD8bpp_ARGB2222, LCD8bpp_BGRA2222, and LCD8bpp_RGBA2222.

7.109.3.29 realX()

Find the real, absolute x coordinate of a point inside a widget with regards to rotation.

Parameters

in	widgetArea	The widget containing the point.
	X	The x coordinate.
	У	The y coordinate.
	rotation	Rotation to perform.

Returns

The absolute x coordinate after applying appropriate rotation.

7.109.3.30 realY()

Find the real, absolute y coordinate of a point inside a widget with regards to rotation.

Parameters

in	widgetArea	The widget containing the point.	
X		The x coordinate.	
	У	The y coordinate.	
	rotation	Rotation to perform.	

Returns

The absolute y coordinate after applying appropriate rotation.

7.109.3.31 rotateRect()

Rotate a rectangle inside another rectangle.

Parameters

in,out	rect The rectangle to rotate.		
	canvas	The rectangle containing the rect to rotate.	
	rotation	Rotation to perform on rect.	

7.109.3.32 stringWidth()

Find string with of the given number of ligatures read from the given TextProvider. After the introduction of Arabic, Thai, Hindi and other languages, ligatures are counted instead of characters. For Latin languages, number of characters equal number of ligatures.

Parameters

in,out	textProvider	The text provider.
	font	The font.
	numChars	Number of characters (ligatures).
	textDirection	The text direction.

Returns

An int16_t.

7.110 LCD16bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD16bpp.hpp>
```

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

• virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

· virtual uint8 t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE INLINE FUNCTION uint16 t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

• static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets green from color.

• static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8 t alpha, uint16 t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf16, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16←
 _t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data←
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blits a 2D source-array to the framebuffer.

static void blitCopyL8 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_ARGB8888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_RGB565 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_RGB888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyAlphaPerPixel (const uint16_t *sourceData, const uint8_t *alphaData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D source-array to the framebuffer.

Static Protected Attributes

static const uint16_t TRANSPARENT_COL = 0xABCD
 Transparency color. Deprecated, do not use.

7.110.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.110.2 Member Function Documentation

```
7.110.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Number of bits per pixel used by the display.

Returns

The number of bits per pixel.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels If true, this data copy contains transparent pixels and require hardware support for		
	to be enabled.	

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
has Transparent Pixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.

Implements LCD.

7.110.2.4 blitCopyAlphaPerPixel()

```
static void blitCopyAlphaPerPixel (
    const uint16_t * sourceData,
    const uint8_t * alphaData,
    const Rect & source,
    const Rect & blitRect,
    uint8_t alpha ) [static], [protected]
```

Blits a 2D source-array to the framebuffer performing alpha-blending per pixel as specified. Always performs a software blend.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 16- bits RGB565 values.	
alphaData	The alpha channel array pointer (points to the beginning of the data)	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.110.2.5 blitCopyARGB8888()

Blits a 2D source-array to the framebuffer perfoming alpha-blending per pixel as specified. If ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 32- bits ARGB8888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.110.2.6 blitCopyL8()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if indexed format is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries.
	by colors criticis.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.110.2.7 blitCopyL8_ARGB8888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored
	as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (ARGB8888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.110.2.8 blitCopyL8_RGB565()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB565 is not supported by the DMA a software blend is performed.

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer points to the beginning of the CLUT color format and size data followed by colors entries stored as 16- bits (RGB565) format. If the source have per pixel alpha channel, then alpha channel data will be following the clut entries data.
source	The location and dimension of the source.

Parameters

blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.110.2.9 blitCopyL8_RGB888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (ARGB8888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.110.2.10 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.110.2.11 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public version of drawGlyph().

Parameters

in	wbuf16	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.110.2.12 drawPartialBitmap()

Draws a portion of a bitmap.

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.110.2.13 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

	T	
dest	The description of where the texture is drawn - can be used to issue a draw off screen.	
gradients	The gradients using in interpolation across the scan line.	
leftEdge	The left edge of the scan line.	
rightEdge	The right edge of the scan line.	
texture	The texture.	
absoluteRect	The containing rectangle in absolute coordinates.	
dirtyAreaAbsolute	The dirty area in absolute coordinates.	
renderVariant	The render variant - includes the algorithm and the pixel format.	
alpha	The alpha.	
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.	

Implements LCD.

7.110.2.14 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.110.2.15 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::RGB565.

Implements LCD.

7.110.2.16 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.110.2.17 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color The 16 bit color value.

Returns

The blue part of the color.

Implements LCD.

7.110.2.18 getBlueFromColor()

Parameters

color	The color.
-------	------------

Returns

The blue from color.

7.110.2.19 getColorFrom24BitRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.110.2.20 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.110.2.21 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.110.2.22 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implements LCD.

7.110.2.23 getGreenFromColor()

color	The color

Returns

The green from color.

7.110.2.24 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

Returns

The red part of the color.

Implements LCD.

7.110.2.25 getRedFromColor()

Parameters

```
color The color.
```

Returns

The red from color.

7.110.2.26 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.110.2.27 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.110.2.28 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.111 LCD16bppSerialFlash Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD16bppSerialFlash.hpp>
```

Public Member Functions

• LCD16bppSerialFlash (FlashDataReader &flashReader)

Creates a LCD16bppSerialFlash object.

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

• virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

· virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8 t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

• static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets green from color.

• static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation=TEXT_ROTATE_0)

Private version of draw-glyph with explicit destination buffer pointer argument.

void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8_

 t alpha)

Blits a 2D source-array to the framebuffer.

void blitCopyL8 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blit←
 Rect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

 void blitCopyL8_ARGB8888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

• void blitCopyL8_RGB565 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

Protected Attributes

· FlashDataReader & flashReader

Flash reader. Used by routines to read pixel data from the flash.

Static Protected Attributes

static const uint16_t TRANSPARENT_COL = 0xABCD
 Transparency color. Deprecated, do not use.

7.111.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.111.2 Constructor & Destructor Documentation

7.111.2.1 LCD16bppSerialFlash()

```
LCD16bppSerialFlash ( {\tt FlashDataReader~\&~flashReader~)}
```

Creates a LCD16bppSerialFlash object. The FlashDataReader object is used to fetch data from the external flash.

in	flashReader	Reference to a FlashDataReader object
----	-------------	---------------------------------------

7.111.3 Member Function Documentation

```
7.111.3.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Returns

The number of bits per pixel.

Number of bits per pixel used by the display.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT COPY WITH ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and tranparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
has Transparent Pixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.

Implements LCD.

7.111.3.4 blitCopyARGB8888()

Blits a 2D source-array to the framebuffer perfoming alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored
	as 32- bits ARGB8888 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.111.3.5 blitCopyL8()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if indexed format is not supported by the DMA a software blend is performed.

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.111.3.6 blitCopyL8_ARGB8888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (ARGB8888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.111.3.7 blitCopyL8_RGB565()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB565 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored
	as 8- bits indexes.
clutData	The source-clut pointer points to the beginning of the CLUT color format and size data followed by colors entries stored as 16- bits (RGB565) format. If the source have per pixel alpha channel, then alpha channel data will be following the clut entries data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.111.3.8 copyFrameBufferRegionToMemory()

```
const Rect & absRegion,
const BitmapId bitmapId ) [virtual]
```

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.111.3.9 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public version of drawGlyph().

in	wbuf16	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	Х	Horizontal offset to start drawing the glyph.

Parameters

У	Vertical offset to start drawing the glyph.
offsetX	Horizontal offset in the glyph to start drawing from.
offsetY	Vertical offset in the glyph to start drawing from.
invalidatedArea	The area to draw within.
glyph	Specifications of the glyph to draw.
glyphData	Data containing the actual glyph (dense format)
dataFormatA4	The glyph is saved using ST A4 format.
color	The color of the glyph.
bitsPerPixel	Bit depth of the glyph.
alpha	The transparency of the glyph.
rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.111.3.10 drawPartialBitmap()

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.111.3.11 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.111.3.12 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.111.3.13 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::RGB565.

Implements LCD.

7.111.3.14 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.111.3.15 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.111.3.16 getBlueFromColor()

Parameters

```
color The color.
```

Returns

The blue from color.

7.111.3.17 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.111.3.18 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.111.3.19 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.111.3.20 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color The 16 bit color va	alue.
---------------------------	-------

Returns

The green part of the color.

Implements LCD.

7.111.3.21 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.111.3.22 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The color value.
-------	------------------

Returns

The red part of the color.

Implements LCD.

7.111.3.23 getRedFromColor()

Parameters

color The color.

Returns

The red from color.

```
7.111.3.24 init()
```

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.111.3.25 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.111.3.26 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.112 LCD1bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles.

```
#include <platform/driver/lcd/LCD1bpp.hpp>
```

Public Member Functions

• virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

 virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the frame buffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8 t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8 t alpha=255)

Draws a filled rectangle in the specified color.

virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

· virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8 t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8 t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE INLINE FUNCTION colortype getColorFromRGB (uint8 t red, uint8 t green, uint8 t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets green from color.

• static FORCE INLINE FUNCTION uint8 t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Not supported for 1bpp.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

virtual void blitCopyRLE (const uint16_t *_sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a run-length encoded 2D source-array to the frame buffer.

void copyRect (const uint8_t *srcAddress, uint16_t srcStride, uint8_t srcPixelOffset, uint8_t *RESTRICT dstAddress, uint16_t dstStride, uint8_t dstPixelOffset, uint16_t width, uint16_t height) const

Copies a rectangular area.

Static Protected Member Functions

• static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

• static void fillMemory (void *RESTRICT dst, colortype color, uint16_t bytesToFill)

Fill memory efficiently.

Additional Inherited Members

7.112.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 1 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.112.2 Member Function Documentation

```
7.112.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
Number of bits per pixel used by the display.
```

Returns

1.

Implements LCD.

Blits a 2D source-array to the frame buffer unless alpha is zero.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must
	be stored as 16-bits RGB565 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (0 = invisible, otherwise solid).
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.112.2.3 blitCopy() [2/2]

Bitmap::BitmapFormat sourceFormat,
const Rect & source,
const Rect & blitRect,
uint8_t alpha,
bool hasTransparentPixels) [virtual]

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT COPY WITH ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	Ignored

Implements LCD.

7.112.2.4 blitCopyRLE()

```
const Rect & blitRect,
uint8_t alpha ) [protected], [virtual]
```

Blits a run-length encoded2D source-array to the frame buffer unless alpha is zero.

Parameters

_sourceData	The source-array pointer (points to the beginning of the data). Data stored in RLE format, where each byte indicates number of pixels with certain color, alternating between black and white. First byte represents black.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (0 = invisible, otherwise solid).

7.112.2.5 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.112.2.6 copyRect()

```
uint16_t srcStride,
uint8_t srcPixelOffset,
uint8_t *RESTRICT dstAddress,
uint16_t dstStride,
uint8_t dstPixelOffset,
uint16_t width,
uint16_t height ) const [protected]
```

Copies a rectangular area.

Parameters

	srcAddress	Source address (byte address).
	srcStride	Source stride (number of bytes to advance to next line).
	srcPixelOffset	Source pixel offset (first pixel in first source byte).
in	dstAddress	If destination address (byte address).
	dstStride	Destination stride (number of bytes to advance to next line).
	dstPixelOffset	Destination pixel offset (first pixel in destination byte).
	width	The width of area (in pixels).
	height	The height of area (in pixels).

7.112.2.7 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

in	wbuf	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.

Parameters

color	The color of the glyph.
bitsPerPixel	Bit depth of the glyph.
alpha	The transparency of the glyph.
rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.112.2.8 drawPartialBitmap()

Draws a portion of a bitmap.

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value (0 = invisible, otherwise solid). Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.112.2.9 drawTextureMapScanLine()

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.

Parameters

texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.112.2.10 fillMemory()

Fill memory efficiently. Try to get 32bit aligned or 16bit aligned and then copy as quickly as possible.

Parameters

out	dst	Pointer to memory to fill.
	color	Color to write to memory, either 0 => 0x00000000 or 1 => 0xFFFFFFF.
	bytesToFill	Number of bytes to fill.

7.112.2.11 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color (values other than 0 or 1 are treated as being 1).
alpha	The rectangle opacity (0 = invisible, otherwise solid). Default is 255 (solid).

Implements LCD.

7.112.2.12 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::BW.

Implements LCD.

7.112.2.13 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.112.2.14 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.112.2.15 getBlueFromColor()

Parameters

color The color.

Returns

The blue from color.

7.112.2.16 getColorFrom24BitRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.112.2.17 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.112.2.18 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.112.2.19 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implements LCD.

7.112.2.20 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.112.2.21 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The color value.
```

Returns

The red part of the color.

Implements LCD.

7.112.2.22 getRedFromColor()

Parameters

color	The color.
-------	------------

Returns

The red from color.

7.112.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.112.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.113 LCD24bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

#include <platform/driver/lcd/LCD24bpp.hpp>

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

virtual uint8 t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

• virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

• static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Gets color from RGB.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

• static FORCE INLINE FUNCTION uint8 t getGreenFromColor (colortype color)

Gets green from color.

• static FORCE INLINE FUNCTION uint8 t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8 t alpha, uint16 t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf16, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16

_t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blits a 2D source-array to the framebuffer.

static void blitCopyL8 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_ARGB8888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_RGB888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

Static Protected Attributes

• static const uint16_t TRANSPARENT_COL = 0xABCD

7.113.1 Detailed Description

Transparency color.

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.113.2 Member Function Documentation

```
7.113.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Returns

The number of bits per pixel.

Number of bits per pixel used by the display.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT COPY WITH ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must
	be stored as 16-bits RGB565 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and tranparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB888 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.113.2.4 blitCopyARGB8888()

Blits a 2D source-array to the framebuffer perfoming alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored
	as 32- bits ARGB8888 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.113.2.5 blitCopyL8()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if indexed format is not supported by the DMA a software blend is performed.

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.113.2.6 blitCopyL8_ARGB8888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (ARGB8888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.113.2.7 blitCopyL8_RGB888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored
	as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed
	by colors entries stored as 32- bits (RGB888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.113.2.8 copyFrameBufferRegionToMemory()

```
const BitmapId bitmapId ) [virtual]
```

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.113.2.9 drawGlyph()

```
void drawGlyph (
        uint16_t * wbuf16,
        Rect widgetArea,
        int16_t x,
        int16_t y,
        uint16_t offsetX,
        uint16_t offsetY,
        const Rect & invalidatedArea,
        const GlyphNode * glyph,
        const uint8_t * glyphData,
        uint8_t dataFormatA4,
        colortype color,
        uint8_t bitsPerPixel,
        uint8_t alpha,
        TextRotation rotation ) [protected], [virtual]
```

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

in	wbuf16	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	Х	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.

Parameters

offsetX	Horizontal offset in the glyph to start drawing from.
offsetY	Vertical offset in the glyph to start drawing from.
invalidatedArea	The area to draw within.
glyph	Specifications of the glyph to draw.
glyphData	Data containing the actual glyph (dense format)
dataFormatA4	The glyph is saved using ST A4 format.
color	The color of the glyph.
bitsPerPixel	Bit depth of the glyph.
alpha	The transparency of the glyph.
rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.113.2.10 drawPartialBitmap()

Draws a portion of a bitmap.

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.113.2.11 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.113.2.12 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.113.2.13 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::RGB888.

Implements LCD.

7.113.2.14 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.113.2.15 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.113.2.16 getBlueFromColor()

Parameters

```
color The color.
```

Returns

The blue from color.

7.113.2.17 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.113.2.18 getColorFromRGB()

Parameters

red	The red.
green	The green.
blue	The blue.

Returns

The color from RGB.

7.113.2.19 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.113.2.20 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Returns

The green part of the color.

Implements LCD.

7.113.2.21 getGreenFromColor()

Parameters

color	The color.
-------	------------

Returns

The green from color.

7.113.2.22 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

	color	The color value.
--	-------	------------------

Returns

The red part of the color.

Implements LCD.

7.113.2.23 getRedFromColor()

Parameters

```
color The color.
```

Returns

The red from color.

```
7.113.2.24 init()
```

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.113.2.25 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.113.2.26 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.114 LCD24DebugPrinter Class Reference

The class LCD24DebugPrinter implements the DebugPrinter interface for printing debug messages on top of 24bit framebuffer.

```
#include <platform/driver/lcd/LCD24bpp.hpp>
```

Public Member Functions

virtual void draw (const LCD &lcd) const

Draws the debug string on top of the framebuffer content.

Additional Inherited Members

7.114.1 Detailed Description

The class LCD24DebugPrinter implements the DebugPrinter interface for printing debug messages on top of 24bit framebuffer.

See also

DebugPrinter

7.114.2 Member Function Documentation

```
7.114.2.1 draw()
```

```
virtual void draw ( {\tt const\ LCD\ \&\ \it lcd\ )\ const\ [virtual]}
```

Draws the debug string on top of the framebuffer content.

Parameters

in	lcd	Reference on the LCD object to use for drawing the debug string.
----	-----	--

Implements DebugPrinter.

7.115 LCD2bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD2bpp.hpp>
```

Public Member Functions

· virtual void init ()

Performs initialization.

• virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

 virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

• virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

virtual uint8 t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

· virtual uint16 t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getRedFromColor (colortype color) const

Gets red from color.

virtual uint8 t getGreenColor (colortype color) const

Gets the green color part of a color.

• virtual uint8 t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets green from color.

static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

void copyRect (const uint8_t *srcAddress, uint16_t srcStride, uint8_t srcPixelOffset, uint8_t *RESTRICT dstAddress, uint16_t dstStride, uint8_t dstPixelOffset, uint16_t width, uint16_t height) const

Copies a rectangular area.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

 static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const uint8_t *sourceAlphaData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

static const uint16_t TRANSPARENT_COL = 0xABCD

Transparency color. Deprecated, do not use.

static const uint8_t alphaTable2bpp [256]

The alpha lookup table to avoid arithmetics when alpha blending.

7.115.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 2 bits per pixel grayscale displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.115.2 Member Function Documentation

```
7.115.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Returns

The number of bits per pixel.

Number of bits per pixel used by the display.

Implements LCD.

```
uint8_t alpha,
bool hasTransparentPixels ) [virtual]
```

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

7.115.2.3 blitCopy() [2/2]

bool hasTransparentPixels) [virtual]

uint8_t alpha,

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD2 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
has Transparent Pixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.

Implements LCD.

7.115.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

Parameters

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 2bpp GRAY2 values.
sourceAlphaData	The alpha channel array pointer (points to the beginning of the data)
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.115.2.5 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.115.2.6 copyRect()

```
uint8_t srcPixelOffset,
uint8_t *RESTRICT dstAddress,
uint16_t dstStride,
uint8_t dstPixelOffset,
uint16_t width,
uint16_t height ) const [protected]
```

Copies a rectangular area.

Parameters

	srcAddress	Source address (byte address).
	srcStride	Source stride (number of bytes to advance to next line).
	srcPixelOffset	Source pixel offset (first pixel in first source byte).
in	dstAddress	If destination address (byte address).
	dstStride	Destination stride (number of bytes to advance to next line).
	dstPixelOffset	Destination pixel offset (first pixel in destination byte).
	width	The width of area (in pixels).
	height	The height of area (in pixels).

7.115.2.7 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

in	wbuf	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	Х	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.

Parameters

color	The color of the glyph.
bitsPerPixel	Bit depth of the glyph.
alpha	The transparency of the glyph.
rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.115.2.8 drawPartialBitmap()

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.115.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.

Parameters

leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.115.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.115.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::GRAY2.

Implements LCD.

7.115.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.115.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The 16 bit color value.
-------	-------------------------

Returns

The blue part of the color.

Implements LCD.

7.115.2.14 getBlueFromColor()

Parameters

```
color The color.
```

Returns

The blue from color.

7.115.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

ſ	red	Value of the red part (0-255).
	green	Value of the green part (0-255).
	blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.115.2.16 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.115.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.115.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implements LCD.

7.115.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.115.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

The color value.	color
------------------	-------

Returns

The red part of the color.

Implements LCD.

7.115.2.21 getRedFromColor()

Parameters

```
color The color.
```

Returns

The red from color.

7.115.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.115.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.115.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.116 LCD32bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD32bpp.hpp>
```

Public Member Functions

· virtual void init ()

Performs initialization.

• virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

 virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8 t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

virtual colortype getColorFrom24BitRGB (uint8 t red, uint8 t green, uint8 t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8 t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

• static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE INLINE FUNCTION uint8 t getGreenFromColor (colortype color)

Gets green from color.

static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyRGB888 (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blits a 2D source-array to the framebuffer.

static void blitCopyRGB565 (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blits a 2D source-array to the framebuffer.

static void blitCopyL8 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_ARGB8888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_RGB888 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8_t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

static void blitCopyL8_RGB565 (const uint8_t *sourceData, const uint8_t *clutData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blits a 2D indexed 8-bit source to the framebuffer.

Additional Inherited Members

7.116.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.116.2 Member Function Documentation

```
7.116.2.1 bitDepth()
```

```
uint8_t bitDepth ( ) const [inline], [virtual]
```

Number of bits per pixel used by the display.

Returns

The number of bits per pixel.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 32-bits ARGB8888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for the	
	to be enabled.	

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and tranparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD32 supports source data formats: RGB565, RGB888 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.	
sourceFormat	The bitmap format used in the source data.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

7.116.2.4 blitCopyL8()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if indexed format is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored	
	as 8- bits indexes.	
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.116.2.5 blitCopyL8_ARGB8888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (ARGB8888) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.116.2.6 blitCopyL8_RGB565()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB565 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored
	as 8- bits indexes.
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 16- bits (RGB565) format.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.116.2.7 blitCopyL8_RGB888()

Blits a 2D indexed 8-bit source to the framebuffer perfoming alpha-blending per pixel as specified if L8_RGB888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-indexes pointer (points to the beginning of the data). The sourceData must be stored as 8- bits indexes.	
clutData	The source-clut pointer (points to the beginning of the CLUT color format and size data followed by colors entries stored as 32- bits (RGB888) format.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.116.2.8 blitCopyRGB565()

Blits a 2D source-array to the framebuffer perfoming alpha-blending per pixel as specified. If! RGB565 is not supported by the DMA a software blend is performed.

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 16- bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	

Parameters

alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)
-------	---

7.116.2.9 blitCopyRGB888()

Blits a 2D source-array to the framebuffer perfoming alpha-blending per pixel as specified. If RGB888 is not supported by the DMA a software blend is performed.

Parameters

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 24- bits RGB888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.116.2.10 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.116.2.11 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public version of drawGlyph().

Parameters

in	wbuf	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.116.2.12 drawPartialBitmap()

```
const Rect & rect,
uint8_t alpha = 255,
bool useOptimized = true ) [virtual]
```

Draws a portion of a bitmap.

Parameters

bitmap	The bitmap to draw.	
X	The absolute x coordinate to place pixel (0, 0) on the screen.	
У	The absolute y coordinate to place pixel (0, 0) on the screen.	
rect	A rectangle describing what region of the bitmap is to be drawn.	
alpha	oha Optional alpha value. Default is 255 (solid).	
useOptimized if false, do not attempt to substitute (parts of) this bitmap with faster fillred		

Implements LCD.

7.116.2.13 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.	
gradients	The gradients using in interpolation across the scan line.	
leftEdge	The left edge of the scan line.	
rightEdge	The right edge of the scan line.	
texture	The texture.	
absoluteRect	The containing rectangle in absolute coordinates.	
dirtyAreaAbsolute	The dirty area in absolute coordinates.	
renderVariant	The render variant - includes the algorithm and the pixel format.	
alpha	The alpha.	
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely	
	perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.	

Implements LCD.

7.116.2.14 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.	
color	olor The rectangle color.	
alpha	The rectangle opacity (255=solid)	

Implements LCD.

7.116.2.15 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::ARGB8888.

Implements LCD.

7.116.2.16 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.116.2.17 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

color	The 16 bit color value.
COIOI	THE TO DIL COIDI VAIUE.

Returns

The blue part of the color.

Implements LCD.

7.116.2.18 getBlueFromColor()

Parameters

color	The color.
-------	------------

Returns

The blue from color.

7.116.2.19 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.116.2.20 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

red	Value of the red part (0-255).
-----	--------------------------------

Parameters

green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color from RGB.

7.116.2.21 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.116.2.22 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The 16 bit color value.

Returns

The green part of the color.

Implements LCD.

7.116.2.23 getGreenFromColor()

Parameters

Returns

The green from color.

7.116.2.24 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

he color value.	color
-----------------	-------

Returns

The red part of the color.

Implements LCD.

7.116.2.25 getRedFromColor()

Parameters

```
color The color.
```

Returns

The red from color.

7.116.2.26 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.116.2.27 nextLine()

Find out how much to advance in the display buffer to get to the next line.

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.116.2.28 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.117 LCD4bpp Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD4bpp.hpp>
```

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

 virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

• virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8 t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8 t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE INLINE FUNCTION colortype getColorFromRGB (uint8 t red, uint8 t green, uint8 t blue)

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

static FORCE INLINE FUNCTION uint8 t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets areen from color.

static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_

t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data

FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

void copyRect (const uint8_t *srcAddress, uint16_t srcStride, uint8_t srcPixelOffset, uint8_t *RESTRICT dstAddress, uint16_t dstStride, uint8_t dstPixelOffset, uint16_t width, uint16_t height) const

Copies a rectangular area.

Static Protected Member Functions

static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

 static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const uint8_t *sourceAlphaData, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

static const uint16_t TRANSPARENT_COL = 0xABCD
 Transparency color. Deprecated, do not use.

7.117.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 4 bits per pixel grayscale displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.117.2 Member Function Documentation

```
7.117.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Number of bits per pixel used by the display.

Returns

The number of bits per pixel.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must
	be stored as 16-bits RGB565 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD4 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.117.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 4bpp GRAY4 values.
sourceAlphaData	The alpha channel array pointer (points to the beginning of the data)
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.117.2.5 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

Implements LCD.

7.117.2.6 copyRect()

Copies a rectangular area.

	srcAddress	Source address (byte address).
	srcStride	Source stride (number of bytes to advance to next line).
	srcPixelOffset	Source pixel offset (first pixel in first source byte).
in	dstAddress	If destination address (byte address).
	dstStride	Destination stride (number of bytes to advance to next line).
	dstPixelOffset	Destination pixel offset (first pixel in destination byte).
	width	The width of area (in pixels).
	height	The height of area (in pixels).

7.117.2.7 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

in	wbuf	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.117.2.8 drawPartialBitmap()

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.

Parameters

У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.117.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The subDivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.117.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.117.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::GRAY4.

Implements LCD.

7.117.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.117.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.117.2.14 getBlueFromColor()

Parameters

Returns

The blue from color.

7.117.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.117.2.16 getColorFromRGB()

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

7.117.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.117.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implements LCD.

7.117.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.117.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color The color value	ıe.
-----------------------	-----

Returns

The red part of the color.

Implements LCD.

7.117.2.21 getRedFromColor()

Parameters

color	The color.
-------	------------

Returns

The red from color.

7.117.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.117.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.117.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.118 LCD8bpp_ABGR2222 Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

```
#include <platform/driver/lcd/LCD8bpp_ABGR2222.hpp>
```

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

 virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8 t bitDepth () const

Number of bits per pixel used by the display.

virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8 t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)
 Gets color from RGB.

static FORCE INLINE FUNCTION uint8 t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets areen from color.

static FORCE_INLINE_FUNCTION uint8_t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf8, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_

t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data

FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

• static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

• static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 t alpha)

Blit a 2D source-array to the framebuffer.

static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

static const uint16_t TRANSPARENT_COL = 0xABCD
 Transparency color. Deprecated, do not use.

7.118.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.118.2 Member Function Documentation

Number of bits per pixel used by the display.

```
7.118.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
```

Returns

8.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that to be enabled.	

Implements LCD.

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.118.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 8-bits ABGR2222 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.118.2.5 blitCopyARGB8888()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 32- bits ARGB8888 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.118.2.6 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.118.2.7 drawGlyph()

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

in	wbuf8	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	Х	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.118.2.8 drawPartialBitmap()

bitmap	The bitmap to draw.	
X	The absolute x coordinate to place pixel $(0, 0)$ on the screen.	

Parameters

У	The absolute y coordinate to place pixel (0, 0) on the screen.	
rect	A rectangle describing what region of the bitmap is to be drawn.	
alpha	Optional alpha value. Default is 255 (solid).	
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.	

Implements LCD.

7.118.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The subDivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.	
gradients	The gradients using in interpolation across the scan line.	
leftEdge	The left edge of the scan line.	
rightEdge	The right edge of the scan line.	
texture	The texture.	
absoluteRect	The containing rectangle in absolute coordinates.	
dirtyAreaAbsolute	The dirty area in absolute coordinates.	
renderVariant	The render variant - includes the algorithm and the pixel format.	
alpha	The alpha.	
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.	

Implements LCD.

7.118.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.118.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::ABGR2222.

Implements LCD.

7.118.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.118.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.118.2.14 getBlueFromColor()

Parameters

color	The color.
-------	------------

Returns

The blue from color.

7.118.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.118.2.16 getColorFromRGB()

Parameters

red	The red.
green	The green.
blue	The blue.

Returns

The color from RGB.

7.118.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.118.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The green part of the color.

Implements LCD.

7.118.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.118.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

color	The color value.
COIOI	THE COIDI Value.

Returns

The red part of the color.

Implements LCD.

7.118.2.21 getRedFromColor()

Parameters

color	The color.
-------	------------

Returns

The red from color.

7.118.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.118.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.118.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.119 LCD8bpp_ARGB2222 Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

#include <platform/driver/lcd/LCD8bpp_ARGB2222.hpp>

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

• virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8 t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE_INLINE_FUNCTION uint16_t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Gets color from RGB.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE_INLINE_FUNCTION uint8_t getGreenFromColor (colortype color)

Gets green from color.

static FORCE INLINE FUNCTION uint8 t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8 t alpha, uint16 t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf8, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

• static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blit a 2D source-array to the framebuffer.

static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

• static const uint16_t TRANSPARENT_COL = 0xABCD

Transparency color. Deprecated, do not use.

7.119.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.119.2 Member Function Documentation

```
7.119.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
Number of bits per pixel used by the display.
```

Returns

8.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

```
const Rect & source,
const Rect & blitRect,
uint8_t alpha,
bool hasTransparentPixels ) [virtual]
```

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and tranparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.119.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

Parameters

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 8-bits ARGB2222 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.119.2.5 blitCopyARGB8888()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 32- bits ARGB8888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.119.2.6 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.119.2.7 drawGlyph()

```
const Rect & invalidatedArea,
const GlyphNode * glyph,
const uint8_t * glyphData,
uint8_t dataFormatA4,
colortype color,
uint8_t bitsPerPixel,
uint8_t alpha,
TextRotation rotation ) [protected], [virtual]
```

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

in	wbuf8	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.119.2.8 drawPartialBitmap()

Parameters

bitmap	The bitmap to draw.	
X	The absolute x coordinate to place pixel (0, 0) on the screen.	
У	The absolute y coordinate to place pixel (0, 0) on the screen.	
rect	3	
alpha		
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.	

Implements LCD.

7.119.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.	
gradients	The gradients using in interpolation across the scan line.	
leftEdge	The left edge of the scan line.	
rightEdge	The right edge of the scan line.	
texture	The texture.	
absoluteRect	The containing rectangle in absolute coordinates.	
dirtyAreaAbsolute	The dirty area in absolute coordinates.	
renderVariant	The render variant - includes the algorithm and the pixel format.	
alpha	The alpha.	
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.	

Implements LCD.

7.119.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.	
color	The rectangle color.	
alpha	The rectangle opacity (255=solid)	

Implements LCD.

7.119.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::ARGB2222.

Implements LCD.

7.119.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.119.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.119.2.14 getBlueFromColor()

Parameters

color The color.

Returns

The blue from color.

7.119.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.119.2.16 getColorFromRGB()

Parameters

red	The red.
green	The green.
blue	The blue.

Returns

The color from RGB.

7.119.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.119.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

Returns

The green part of the color.

Implements LCD.

7.119.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.119.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

```
color The color value.
```

Returns

The red part of the color.

Implements LCD.

7.119.2.21 getRedFromColor()

Parameters

Returns

The red from color.

7.119.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.119.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.119.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.120 LCD8bpp_BGRA2222 Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

#include <platform/driver/lcd/LCD8bpp_BGRA2222.hpp>

Public Member Functions

· virtual void init ()

Performs initialization.

 virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

• virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

· virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8_t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE INLINE FUNCTION uint16 t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Gets color from RGB.

• static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

• static FORCE INLINE FUNCTION uint8 t getGreenFromColor (colortype color)

Gets green from color.

static FORCE INLINE FUNCTION uint8 t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf8, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

• static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blit a 2D source-array to the framebuffer.

static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

• static const uint16_t TRANSPARENT_COL = 0xABCD

Transparency color. Deprecated, do not use.

7.120.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.120.2 Member Function Documentation

```
7.120.2.1 bitDepth()

uint8_t bitDepth ( ) const [inline], [virtual]

Number of bits per pixel used by the display.

Returns
```

8.

Implements LCD.

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

```
const Rect & source,
const Rect & blitRect,
uint8_t alpha,
bool hasTransparentPixels ) [virtual]
```

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.
sourceFormat	The bitmap format used in the source data.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending (255 = solid, no blending)
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that
	to be enabled.

Implements LCD.

7.120.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

Parameters

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 8-bits BGRA2222 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.120.2.5 blitCopyARGB8888()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored	
	as 32- bits ARGB8888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.120.2.6 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.120.2.7 drawGlyph()

```
const Rect & invalidatedArea,
const GlyphNode * glyph,
const uint8_t * glyphData,
uint8_t dataFormatA4,
colortype color,
uint8_t bitsPerPixel,
uint8_t alpha,
TextRotation rotation ) [protected], [virtual]
```

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

in	wbuf8	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.0
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.120.2.8 drawPartialBitmap()

Parameters

bitmap	The bitmap to draw.
X	The absolute x coordinate to place pixel (0, 0) on the screen.
У	The absolute y coordinate to place pixel (0, 0) on the screen.
rect	A rectangle describing what region of the bitmap is to be drawn.
alpha	Optional alpha value. Default is 255 (solid).
useOptimized	if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.

Implements LCD.

7.120.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.
gradients	The gradients using in interpolation across the scan line.
leftEdge	The left edge of the scan line.
rightEdge	The right edge of the scan line.
texture	The texture.
absoluteRect	The containing rectangle in absolute coordinates.
dirtyAreaAbsolute	The dirty area in absolute coordinates.
renderVariant	The render variant - includes the algorithm and the pixel format.
alpha	The alpha.
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.

Implements LCD.

7.120.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.
color	The rectangle color.
alpha	The rectangle opacity (255=solid)

Implements LCD.

7.120.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::BGRA2222.

Implements LCD.

7.120.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.120.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.120.2.14 getBlueFromColor()

Parameters

color The color.

Returns

The blue from color.

7.120.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).
green	Value of the green part (0-255).
blue	Value of the blue part (0-255).

Returns

The color representation depending on LCD color format.

Implements LCD.

7.120.2.16 getColorFromRGB()

Parameters

red	The red.
green	The green.
blue	The blue.

Returns

The color from RGB.

7.120.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.120.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

Returns

The green part of the color.

Implements LCD.

7.120.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.120.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

color	The color value.

Returns

The red part of the color.

Implements LCD.

7.120.2.21 getRedFromColor()

Parameters

color	The color.
-------	------------

Returns

The red from color.

7.120.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.120.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.120.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?	
textRotation	Rotation to perform.	

Returns

How much to advance to get to the next pixel.

7.121 LCD8bpp_RGBA2222 Class Reference

This class contains the various low-level drawing routines for drawing bitmaps.

#include <platform/driver/lcd/LCD8bpp_RGBA2222.hpp>

Public Member Functions

· virtual void init ()

Performs initialization.

• virtual void drawPartialBitmap (const Bitmap &bitmap, int16_t x, int16_t y, const Rect &rect, uint8_t alpha=255, bool useOptimized=true)

Draws a portion of a bitmap.

virtual void blitCopy (const uint16_t *sourceData, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer.

virtual void blitCopy (const uint8_t *sourceData, Bitmap::BitmapFormat sourceFormat, const Rect &source, const Rect &blitRect, uint8_t alpha, bool hasTransparentPixels)

Blits a 2D source-array to the framebuffer while converting the format.

 virtual uint16_t * copyFrameBufferRegionToMemory (const Rect &visRegion, const Rect &absRegion, const BitmapId bitmapId)

Copies part of the frame buffer region to memory.

virtual void fillRect (const Rect &rect, colortype color, uint8_t alpha=255)

Draws a filled rectangle in the specified color.

• virtual uint8_t bitDepth () const

Number of bits per pixel used by the display.

· virtual Bitmap::BitmapFormat framebufferFormat () const

Framebuffer format used by the display.

• virtual uint16_t framebufferStride () const

Framebuffer stride in bytes.

• virtual colortype getColorFrom24BitRGB (uint8_t red, uint8_t green, uint8_t blue) const

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

virtual uint8_t getRedColor (colortype color) const

Gets the red color part of a color.

virtual uint8_t getGreenColor (colortype color) const

Gets the green color part of a color.

virtual uint8 t getBlueColor (colortype color) const

Gets the blue color part of a color.

Static Public Member Functions

static FORCE INLINE FUNCTION uint16 t getFramebufferStride ()

Framebuffer stride in bytes.

• static FORCE_INLINE_FUNCTION colortype getColorFromRGB (uint8_t red, uint8_t green, uint8_t blue)

Gets color from RGB.

static FORCE_INLINE_FUNCTION uint8_t getRedFromColor (colortype color)

Gets red from color.

static FORCE INLINE FUNCTION uint8 t getGreenFromColor (colortype color)

Gets green from color.

static FORCE INLINE FUNCTION uint8 t getBlueFromColor (colortype color)

Gets blue from color.

Protected Member Functions

 virtual void drawTextureMapScanLine (const DrawingSurface &dest, const Gradients &gradients, const Edge *leftEdge, const Edge *rightEdge, const TextureSurface &texture, const Rect &absoluteRect, const Rect &dirtyAreaAbsolute, RenderingVariant renderVariant, uint8_t alpha, uint16_t subDivisionSize)

Draw scan line. Draw one horizontal line of the texture map on screen. The scan line will be drawn using perspective correct texture mapping. The appearance of the line is determined by the left and right edge and the gradients structure. The edges contain the information about the x,y,z coordinates of the left and right side respectively and also information about the u,v coordinates of the texture map used. The gradients structure contains information about how to interpolate all the values across the scan line. The data drawn should be present in the texture argument.

virtual void drawGlyph (uint16_t *wbuf8, Rect widgetArea, int16_t x, int16_t y, uint16_t offsetX, uint16_
 t offsetY, const Rect &invalidatedArea, const GlyphNode *glyph, const uint8_t *glyphData, uint8_t data
 FormatA4, colortype color, uint8_t bitsPerPixel, uint8_t alpha, TextRotation rotation)

Private version of draw-glyph with explicit destination buffer pointer argument.

Static Protected Member Functions

• static int nextPixel (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next pixel.

static int nextLine (bool rotatedDisplay, TextRotation textRotation)

Find out how much to advance in the display buffer to get to the next line.

static void blitCopyARGB8888 (const uint32_t *sourceData, const Rect &source, const Rect &blitRect, uint8
 _t alpha)

Blit a 2D source-array to the framebuffer.

static void blitCopyAlphaPerPixel (const uint16_t *sourceData16, const Rect &source, const Rect &blitRect, uint8 t alpha)

Blit a 2D source-array to the framebuffer.

Static Protected Attributes

• static const uint16_t TRANSPARENT_COL = 0xABCD

Transparency color. Deprecated, do not use.

7.121.1 Detailed Description

This class contains the various low-level drawing routines for drawing bitmaps, texts and rectangles on 16 bits per pixel displays.

Note

All coordinates are expected to be in absolute coordinates!

See also

LCD

7.121.2 Member Function Documentation

```
7.121.2.1 bitDepth()
uint8_t bitDepth ( ) const [inline], [virtual]
Number of bits per pixel used by the display.
```

Returns

8.

Implements LCD.

bool hasTransparentPixels) [virtual]

Blits a 2D source-array to the framebuffer performing alpha-blending (and transparency keying) as specified Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must	
	be stored as 16-bits RGB565 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

```
const Rect & source,
const Rect & blitRect,
uint8_t alpha,
bool hasTransparentPixels ) [virtual]
```

Blits a 2D source-array to the framebuffer perfoming alpha-blending (and transparency keying) as specified. Performs a software blend if HAL does not support BLIT_COPY_WITH_ALPHA and alpha != 255. LCD16 supports source data formats: RGB565 and ARGB8888.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored in a format suitable for the selected display.	
sourceFormat	The bitmap format used in the source data.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending (255 = solid, no blending)	
hasTransparentPixels	If true, this data copy contains transparent pixels and require hardware support for that	
	to be enabled.	

Implements LCD.

7.121.2.4 blitCopyAlphaPerPixel()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified Performs always a software blend.

Parameters

sourceData16	The source-array pointer (points to the beginning of the data). The sourceData must be stored as 8-bit RGBA2222 values.
source	The location and dimension of the source.
blitRect	A rectangle describing what region is to be drawn.
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)

7.121.2.5 blitCopyARGB8888()

Blit a 2D source-array to the framebuffer performing alpha-blending per pixel as specified if ARGB8888 is not supported by the DMA a software blend is performed.

Parameters

sourceData	The source-array pointer (points to the beginning of the data). The sourceData must be stored	
	as 32- bits ARGB8888 values.	
source	The location and dimension of the source.	
blitRect	A rectangle describing what region is to be drawn.	
alpha	The alpha value to use for blending applied to the whole image (255 = solid, no blending)	

7.121.2.6 copyFrameBufferRegionToMemory()

Copies part of the framebuffer region to memory. The memory is given as Bitmapld, which can be BITMAP_ANI ← MATION_STORAGE. The two regions given are the visible region and the absolute region on screen. This is used to copy only a part of an area. This might be the case if a SnapshotWidget is placed inside a Container where parts of the SnapshowWidget is outside the area defined by the Container. The visible region must be completely inside the absolute region.

Note

There is only one instance of animation storage. The content of the animation storage outside the given region is undefined.

Parameters

visRegion	The visible region.
absRegion	The absolute region.
bitmapld	Identifier for the bitmap.

Returns

Null if it fails, else a pointer to the data in the given bitmap.

See also

blitCopy

Implements LCD.

7.121.2.7 drawGlyph()

```
const Rect & invalidatedArea,
const GlyphNode * glyph,
const uint8_t * glyphData,
uint8_t dataFormatA4,
colortype color,
uint8_t bitsPerPixel,
uint8_t alpha,
TextRotation rotation ) [protected], [virtual]
```

Private version of draw-glyph with explicit destination buffer pointer argument. For all parameters (except the buffer pointer) see the public function drawString().

Parameters

in	wbuf8	The destination (frame) buffer to draw to.
	widgetArea	The canvas to draw the glyph inside.
	X	Horizontal offset to start drawing the glyph.
	У	Vertical offset to start drawing the glyph.
	offsetX	Horizontal offset in the glyph to start drawing from.
	offsetY	Vertical offset in the glyph to start drawing from.
	invalidatedArea	The area to draw within.
	glyph	Specifications of the glyph to draw.
	glyphData	Data containing the actual glyph (dense format)
	dataFormatA4	The glyph is saved using ST A4 format.
	color	The color of the glyph.
	bitsPerPixel	Bit depth of the glyph.
	alpha	The transparency of the glyph.
	rotation	Rotation to do before drawing the glyph.

Implements LCD.

7.121.2.8 drawPartialBitmap()

Parameters

bitmap	The bitmap to draw.	
X	The absolute x coordinate to place pixel (0, 0) on the screen.	
У	The absolute y coordinate to place pixel (0, 0) on the screen.	
rect	A rectangle describing what region of the bitmap is to be drawn.	
alpha	Optional alpha value. Default is 255 (solid).	
useOptimized	ized if false, do not attempt to substitute (parts of) this bitmap with faster fillrects.	

Implements LCD.

7.121.2.9 drawTextureMapScanLine()

The scan line will be drawn using the additional arguments. The scan line will be placed and clipped using the absolute and dirty rectangles The alpha will determine how the scan line should be alpha blended. The sub⇔ DivisionSize will determine the size of the piecewise affine texture mapped lines.

Parameters

dest	The description of where the texture is drawn - can be used to issue a draw off screen.	
gradients	The gradients using in interpolation across the scan line.	
leftEdge	The left edge of the scan line.	
rightEdge	The right edge of the scan line.	
texture	The texture.	
absoluteRect	The containing rectangle in absolute coordinates.	
dirtyAreaAbsolute	The dirty area in absolute coordinates.	
renderVariant	The render variant - includes the algorithm and the pixel format.	
alpha	The alpha.	
subDivisionSize	The size of the subdivisions of the scan line. A value of 1 will give a completely perspective correct texture mapped scan line. A large value will give an affine texture mapped scan line.	

Implements LCD.

7.121.2.10 fillRect()

Draws a filled rectangle in the specified color.

Parameters

rect	The rectangle to draw in absolute coordinates.	
color	The rectangle color.	
alpha	The rectangle opacity (255=solid)	

Implements LCD.

7.121.2.11 framebufferFormat()

```
Bitmap::BitmapFormat framebufferFormat ( ) const [inline], [virtual]
```

Framebuffer format used by the display

Returns

Bitmap::RGBA2222.

Implements LCD.

7.121.2.12 framebufferStride()

```
uint16_t framebufferStride ( ) const [inline], [virtual]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

Implements LCD.

7.121.2.13 getBlueColor()

Gets the blue color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The 16 bit color value.
```

Returns

The blue part of the color.

Implements LCD.

7.121.2.14 getBlueFromColor()

Parameters

color The color.

Returns

The blue from color.

7.121.2.15 getColorFrom24BitRGB()

```
colortype getColorFrom24BitRGB (
          uint8_t red,
          uint8_t green,
          uint8_t blue ) const [inline], [virtual]
```

Generates a color representation to be used on the LCD, based on 24 bit RGB values.

Parameters

red	Value of the red part (0-255).	
green	Value of the green part (0-255).	
blue	Value of the blue part (0-255).	

Returns

The color representation depending on LCD color format.

Implements LCD.

7.121.2.16 getColorFromRGB()

Parameters

red	The red.
green	The green.
blue	The blue.

Returns

The color from RGB.

7.121.2.17 getFramebufferStride()

```
FORCE_INLINE_FUNCTION static uint16_t getFramebufferStride ( ) [inline], [static]
```

Framebuffer stride in bytes. The distance (in bytes) from the start of one framebuffer row, to the next.

Returns

The number of bytes in one framebuffer row.

7.121.2.18 getGreenColor()

Gets the green color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

Returns

The green part of the color.

Implements LCD.

7.121.2.19 getGreenFromColor()

Parameters

```
color The color.
```

Returns

The green from color.

7.121.2.20 getRedColor()

Gets the red color part of a color. As this function must work for all color depths, it can be somewhat slow if used in speed critical sections. Consider finding the color in another way, if possible.

Parameters

```
color The color value.
```

Returns

The red part of the color.

Implements LCD.

7.121.2.21 getRedFromColor()

Parameters

Returns

The red from color.

7.121.2.22 init()

```
void init ( ) [virtual]
```

Performs initialization.

Reimplemented from LCD.

7.121.2.23 nextLine()

Find out how much to advance in the display buffer to get to the next line.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next line.

7.121.2.24 nextPixel()

Find out how much to advance in the display buffer to get to the next pixel.

Parameters

rotatedDisplay	Is the display running in portrait mode?
textRotation	Rotation to perform.

Returns

How much to advance to get to the next pixel.

7.122 LED Class Reference

A led.

```
#include <touchgfx/hal/LED.hpp>
```

Static Public Member Functions

• static void init ()

Perform configuration of IO pins.

• static void on (uint8_t nr)

Turn on a LED.

static void off (uint8_t nr)

Turn off a LED.

• static void toggle (uint8_t nr)

Toggles a LED.

• static bool get (uint8_t nr)

Get state of a LED.

7.122.1 Member Function Documentation

```
7.122.1.1 get()
```

```
static bool get (
            uint8_t nr ) [static]
```

Get state of a LED.

Parameters

```
nr of the LED to toggle.
```

Returns

the state of the LED.

7.122.1.5 toggle()

```
static void toggle ( \mbox{uint8\_t } nr \mbox{ ) [static]} \label{eq:uint8_toggle}
```

Toggles a LED.

Parameters

nr of the LED to toggle.

7.123 Line Class Reference

Simple CanvasWidget capable of drawing a line.

```
#include <touchgfx/widgets/canvas/Line.hpp>
```

Public Types

• enum LINE_ENDING_STYLE { BUTT_CAP_ENDING, ROUND_CAP_ENDING, SQUARE_CAP_ENDING } Values that represent line ending styles.

Public Member Functions

• Line () Construct a new Line. template<typename T > void setLine (T x1, T y1, T x2, T y2) Sets the endpoints of the line. • template<typename T >void setStart (T x, T y) Sets the start point for this Line. void setStart (CWRUtil::Q5 xQ5, CWRUtil::Q5 yQ5) Sets the start point for this Line. • template<typename T > void updateStart (T x, T y) Update the start point for this Line. void updateStart (CWRUtil::Q5 xQ5, CWRUtil::Q5 yQ5) Update the start point for this Line. template<typename T > void getStart (T &x, T &y) const Gets the start coordinates for the line. template<typename T > void setEnd (T x, T y) Sets the end point for this Line. void setEnd (CWRUtil::Q5 xQ5, CWRUtil::Q5 yQ5) Sets the end point for this Line. • template<typename T > void updateEnd (T x, T y) Update the end point for this Line. void updateEnd (CWRUtil::Q5 xQ5, CWRUtil::Q5 yQ5) Update the end point for this Line. template<typename T > void getEnd (T &x, T &y) const Gets the end coordinates for the line. • template<typename T >void setLineWidth (T width) Sets the width for this Line. void setLineWidth (CWRUtil::Q5 widthQ5) Sets the width for this Line. • template<typename T >void updateLineWidth (T width) Update the width for this Line. void updateLineWidth (CWRUtil::Q5 widthQ5) Update the width for this Line. • template<typename T > void getLineWidth (T &width) const Gets line width. • template<typename T >T getLineWidth () const Gets line width.

TouchGFX v4.12.3 (C)opyright STMicroelectronics 2014-2019

Sets line ending style.

void setLineEndingStyle (LINE_ENDING_STYLE lineEnding)

LINE_ENDING_STYLE getLineEndingStyle () const

Gets line ending style.

void setCapPrecision (int precision)

Sets a precision of the ends of the Line.

• virtual bool drawCanvasWidget (const Rect &invalidatedArea) const

Draws the Line.

· virtual Rect getMinimalRect () const

Gets minimal rectangle containing the shape drawn by this widget.

void updateLengthAndAngle (CWRUtil::Q5 length, CWRUtil::Q5 angle)

Update the end point for this Line.

Additional Inherited Members

7.123.1 Detailed Description

Simple CanvasWidget capable of drawing a line from one point to another point. The end points can be moved to new locations and the line width can easily be set and changed. A 10 pixel long line along the top of the screen with a width on 1 pixel has endpoints in (0, 0.5) and (10, 0.5) and line width 1. The Line class calculates the corners of the shape, which in this case would be (0, 0), (10, 0), (10, 1) and (0, 1) and tells CWR to moveTo the first coordinate and then lineTo the next coordinates in order. Finally it tells CWR to render the inside of the shape using a Painter object.

The Line class caches the four corners of the shape to speed up redrawing. In general, drawing lines involve some extra mathematics for calculating the normal vector of the line and this computation would slow down re-draws if not cached.

See also

CanvasWidget

7.123.2 Member Enumeration Documentation

7.123.2.1 LINE ENDING STYLE

enum LINE_ENDING_STYLE

Enumerator

BUTT_CAP_ENDING	The line ending is cut 90 degrees at the end of the line.
ROUND_CAP_ENDING	The line ending is rounded as a circle with center at the end of the line.
SQUARE_CAP_ENDING	The line ending is cut 90 degrees, but extends half the width of the line.

7.123.3 Constructor & Destructor Documentation

7.123.3.1 Line()

Line ()

Construct a new line.

7.123 Line Class Reference 589

7.123.4 Member Function Documentation

7.123.4.1 drawCanvasWidget()

Draws the Line. This class supports partial drawing, so only the area described by the rectangle will be drawn. As the corners of the shape are cached, the line can quickly be redrawn when required.

Parameters

The rectangle to draw, with coo	ordinates relative to this drawable.
---------------------------------	--------------------------------------

Returns

true if it succeeds, false if it fails.

Implements CanvasWidget.

7.123.4.2 getEnd()

```
template< typename T > void getEnd (  \mbox{T \& } x,   \mbox{T \& } y \; ) \; \mbox{const [inline]}
```

Gets the end coordinates for the line.

Template Parameters

Τ	Generic type parameter, either int or float.

Parameters

out	X	The x coordinate.
out	у	The y coordinate.

See also

setEnd setLine

7.123.4.3 getLineEndingStyle()

```
{\tt LINE\_ENDING\_STYLE} \ \ {\tt getLineEndingStyle} \ \ (\ ) \ \ {\tt const}
```

Gets line ending style. See LINE_ENDING_STYLE for possible styles.

Returns

The line ending style.

See also

LINE_ENDING_STYLE setLineEndingStyle

```
7.123.4.4 getLineWidth() [1/2]
```

```
template< typename T > void getLineWidth ( T & width ) const [inline]
```

Gets line width.

Template Parameters

T Generic type parameter, either int or float.

Parameters

out <i>width</i>	The line width.
------------------	-----------------

See also

setLineWidth

```
7.123.4.5 getLineWidth() [2/2]
```

```
template< typename T > T getLineWidth ( ) const [inline]
```

Gets line width.

Template Parameters

```
T Generic type parameter, either int (default) or float.
```

Returns

The line width.

See also

setLineWidth

param [out] width The line width.

7.123.4.6 getMinimalRect()

```
Rect getMinimalRect ( ) const [virtual]
```

7.123 Line Class Reference 591

Gets minimal rectangle containing the shape drawn by this widget. Default implementation returns the size of the entire widget, but this function should be overwritten in subclasses and return the minimal rectangle containing the shape. See classes such as Circle for example implementations.

Returns

The minimal rectangle containing the shape drawn by this widget.

Reimplemented from CanvasWidget.

7.123.4.7 getStart()

Gets the start coordinates for the line.

Template Parameters

T	Generic type parameter, either int or float.
---	--

Parameters

out	Χ	The x coordinate.
out	У	The y coordinate.

See also

setStart setLine

7.123.4.8 setCapPrecision()

Sets a precision of the ends of the Line arc. The precision is given in degrees where 18 is the default which results in a nice half circle with 10 line segments. 90 will draw "an arrow head".

Note

The line is not invalidated.

This is only used if line ending is set to ROUND_CAP_ENDING.

Parameters

precision	The new cap precision.

7.123.4.9 setEnd() [1/2]

```
template< typename T > void setEnd ( T x, T y ) [inline]
```

Sets the end point for this Line.

Note

The area containing the Line is not invalidated.

Template Parameters

T Generic type parameter, either int or flo	oat.
---	------

Parameters

X	The x coordinate of the end point.
У	The y coordinate of the end point.

See also

updateEnd getEnd

Sets the end point for this Line.

Parameters

хG)5	The x coordinate of the end point in Q5 format.
уG)5	The y coordinate of the end point in Q5 format.

Note

The area containing the Line is not invalidated.

See also

updateEnd getEnd

7.123.4.11 setLine()

```
template< typename T > void setLine ( T x1,
```

```
T y1,
T x2,
T y2) [inline]
```

Sets the endpoints of the line.

Note

The area containing the Line is not invalidated.

Template Parameters

	Т	Generic type parameter, either int or float.
۱	'	Generic type parameter, either int or noat.

Parameters

x1	The x coordinate of the start point.
y1	The y coordinate of the start point.
x2	The x coordinate of the end point.
y2	The y coordinate of the end point.

See also

setStart setEnd

7.123.4.12 setLineEndingStyle()

Sets line ending style. See LINE_ENDING_STYLE for possible styles.

Note

The area containing the Line is not invalidated.

Parameters

lineEnding	The line ending style.
------------	------------------------

See also

```
LINE_ENDING_STYLE getLineEndingStyle
```

```
7.123.4.13 setLineWidth() [1/2]
```

Sets the width for this Line.

Note

The area containing the Line is not invalidated.

Template Parameters

Generic type parameter, either int or float.

Parameters

width The width of the line measured in pixels.

See also

updateLineWidth

Sets the width for this Line.

Parameters

widthQ5 The width of the line measured in pixels in Q5 format.

Note

The area containing the Line is not invalidated.

See also

updateLineWidth

```
7.123.4.15 setStart() [1/2] template< typename T > void setStart ( T x, T y ) [inline]
```

Sets the start point for this Line.

Note

The area containing the Line is not invalidated.

Template Parameters

T | Generic type parameter, either int or float.

7.123 Line Class Reference 595

Parameters

X	The x coordinate of the start point.
У	The y coordinate of the start point.

See also

```
updateStart
getStart
setLine
setEnd
```


Sets the start point for this Line.

Parameters

xQ5	The x coordinate of the start point in Q5 format.
yQ5	The y coordinate of the start point in Q5 format.

Note

The area containing the Line is not invalidated.

See also

```
updateStart
getStart
setLine
setEnd
```

```
7.123.4.17 updateEnd() [1/2]
```

Update the end point for this Line. The rectangle that surrounds the line before and after will be invalidated.

Note

The area containing the Line is invalidated before and after the change.

Template Parameters

Parameters

L		The x coordinate of the end point.
	У	The y coordinate of the end point.

Update the end point for this Line. The rectangle that surrounds the line before and after will be invalidated.

Parameters

xQ5	The x coordinate of the end point in Q5 format.
yQ5	The y coordinate of the end point in Q5 format.

Note

The area containing the Line is invalidated before and after the change.

7.123.4.19 updateLengthAndAngle()

Update the end point for this Line given the new length and angle. The rectangle that surrounds the line before and after will be invalidated.

Parameters

length	The new length of the line in Q5 format.
angle	The new angle of the line in Q5 format.

Note

The area containing the Line is invalidated before and after the change.

Update the width for this Line and invalidates the minimal rectangle surrounding the line on screen.

Note

The area containing the Line is invalidated before and after the change.

7.123 Line Class Reference 597

Template Parameters

Generic type parameter, either int or float.

Parameters

width	The width of the line measured in pixels.
-------	---

See also

setLineWidth

Update the width for this Line and invalidates the minimal rectangle surrounding the line on screen.

Parameters

	widthQ5	The width of the line measured in pixels in Q5 format.
--	---------	--

Note

The area containing the Line is invalidated before and after the change.

See also

setLineWidth

7.123.4.22 updateStart() [1/2]

```
template< typename T > void updateStart ( T x, T y ) [inline]
```

Update the start point for this Line. The rectangle that surrounds the line before and after will be invalidated.

Note

The area containing the Line is invalidated before and after the change.

Template Parameters

T Generic type parameter, either int or float.

Parameters

x The x coordinate of the start point.

Parameters

```
y The y coordinate of the start point.
```

See also

```
setStart
updateEnd
```

7.123.4.23 updateStart() [2/2]

Update the start point for this Line. The rectangle that surrounds the line before and after will be invalidated.

Parameters

xQ5	The x coordinate of the start point in Q5 format.
yQ5	The y coordinate of the start point in Q5 format.

Note

The area containing the Line is invalidated before and after the change.

See also

setStart updateEnd

7.124 LineProgress Class Reference

A line progress.

```
#include <touchgfx/containers/progress_indicators/LineProgress.hpp>
```

Public Member Functions

• LineProgress ()

Default constructor.

virtual ~LineProgress ()

Destructor.

virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the position and dimension of the line progress indicator.

virtual void setPainter (AbstractPainter &painter)

Sets a painter.

virtual void setStart (int x, int y)

Sets a starting point for the line.

· virtual void getStart (int &x, int &y) const

Gets the coordinates of the starting point of the line.

virtual void setEnd (int x, int y)

Sets an end point for the line.

virtual void getEnd (int &x, int &y) const

Gets the coordinates of the end point of the line.

virtual void setLineWidth (int width)

Sets line width.

• virtual int getLineWidth () const

Gets line width.

• virtual void setLineEndingStyle (Line::LINE ENDING STYLE lineEndingStyle)

Sets line ending style.

 $\bullet \ \ virtual \ Line:: LINE_ENDING_STYLE \ getLineEndingStyle \ () \ const$

Gets line ending style.

virtual void setAlpha (uint8_t alpha)

Sets the alpha.

virtual uint8_t getAlpha () const

Gets the alpha.

virtual void setValue (int value)

Sets a value.

Protected Attributes

· Line line

The line.

CWRUtil::Q5 startX

The start x coordinate.

CWRUtil::Q5 startY

The start y coordinate.

• CWRUtil::Q5 endX

The end x coordinate.

CWRUtil::Q5 endY

The end y coordinate.

Additional Inherited Members

7.124.1 Detailed Description

Using Line from canvas widgets, progress will be rendered as a line. The line does not need to horizontal or vertical, but can start at any coordinate and finish at any coordinate.

Note

As LineProgress uses CanvasWidgetRenderer, it is important that a buffer is set up by calling CanvasWidget← Rendere::setBuffer().

7.124.2 Constructor & Destructor Documentation

7.124.2.1 LineProgress()

LineProgress ()

Default constructor.

7.124.2.2 \sim LineProgress()

```
~LineProgress ( ) [virtual]
```

Destructor.

7.124.3 Member Function Documentation

```
7.124.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const [virtual]
```

Gets the alpha.

Returns

The alpha.

See also

Line::getAlpha

7.124.3.2 getEnd()

Gets the coordinates of the end point of the line. Beware that this is not the coordinates of the current progress of the line, but the coordinates when the line is at 100%.

Parameters

out	Х	The x coordinate.
out	У	The y coordinate.

7.124.3.3 getLineEndingStyle()

```
Line::LINE_ENDING_STYLE getLineEndingStyle ( ) const [virtual]
```

Gets line ending style.

Returns

The line ending style.

7.124.3.4 getLineWidth()

```
int getLineWidth ( ) const [virtual]
```

Gets line width.

Returns

The line width.

7.124.3.5 getStart()

Gets the coordinates of the starting point of the line.

Parameters

out	Х	The x coordinate.
out	У	The y coordinate.

7.124.3.6 setAlpha()

Sets the alpha for the line.

Parameters

alpha The alpha.

See also

Line::setAlpha

7.124.3.7 setEnd()

Sets an end point for the line. When progress is at 100%, the line will go from the coordinates set by setStart() to these coordinates set by setEnd()

Parameters

λ	(The x coordinate of the end point.
y	/	The y coordinate of the end point.

See also

setStart

7.124.3.8 setLineEndingStyle()

Sets line ending style.

Parameters

lineEndingStyle	The line ending style.

See also

Line::setLineEndingStyle

7.124.3.9 setLineWidth()

Sets line width.

Parameters

```
width The width.
```

See also

Line::setLineWidth

7.124.3.10 setPainter()

Sets a painter to be used for drawing the line. This can be any Painter, a simple single color painter, a bitmap painter or a custom painter.

Parameters

in	painter	The painter.

7.124.3.11 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the line progress indicator relative to the background image.

Parameters

X	The x coordinate.	
У	The y coordinate.	
width	The width of the line progress indicator.	
height	The height of the line progress indicator.	

Reimplemented from AbstractProgressIndicator.

7.124.3.12 setStart()

```
void setStart (  \mbox{int } x, \\ \mbox{int } y \;) \quad \mbox{[virtual]}
```

Sets a starting point for the line.

Parameters

X	The x coordinate of the start point.	
У	The y coordinate of the start poin	

See also

setEnd

7.124.3.13 setValue()

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

value	The value.

Reimplemented from AbstractProgressIndicator.

7.125 ListLayout Class Reference

This class provides a layout mechanism for arranging Drawable instances adjacently in the specified Direction.

```
#include <touchgfx/containers/ListLayout.hpp>
```

Public Member Functions

ListLayout (const Direction d=SOUTH)

Constructor.

virtual ~ListLayout ()

Destructor.

virtual void setDirection (const Direction d)

Sets the direction of the ListLayout.

virtual Direction getDirection () const

Gets the direction of the ListLayout.

• virtual void add (Drawable &d)

Adds a Drawable instance to the end of the list.

• virtual void remove (Drawable &d)

Removes a Drawable.

• virtual void insert (Drawable *previousElement, Drawable &d)

Inserts a Drawable.

• virtual void removeAll ()

Removes all children.

virtual uint16_t getType () const

For GUI testing only.

Additional Inherited Members

7.125.1 Detailed Description

This class provides a layout mechanism for arranging Drawable instances adjacently in the specified Direction. The first element in the ListLayout is positioned in the ListLayout origin (0,0). The dimension of this class is automatically expanded to cover the area of the added Drawable instances.

See also

Container

7.125.2 Constructor & Destructor Documentation

7.125.2.1 ListLayout()

```
ListLayout ( const Direction d = SOUTH ) [inline]
```

Constructor. Constructs a ListLayout instance that arranges the added elements in the specified Direction.

Parameters

d The direction to grow in when adding children.

```
7.125.2.2 ∼ListLayout()
```

```
~ListLayout ( ) [inline], [virtual]
```

Destructor.

7.125.3 Member Function Documentation

```
7.125.3.1 add() \label{eq:condition} \mbox{void add (} \mbox{$\tt Drawable \& $d$ }) \mbox{ [virtual]}
```

Adds a Drawable instance to the end of the list. The Drawable dimensions shall be set prior to addition.

Parameters

in	d	The Drawable to add.
----	---	----------------------

Reimplemented from Container.

7.125.3.2 getDirection()

```
Direction getDirection ( ) const [inline], [virtual]
```

Gets the direction of the ListLayout.

Returns

The current direction to grow in when added children (either SOUTH or EAST).

See also

setDirection()

```
7.125.3.3 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_LISTLAYOUT.

Reimplemented from Container.

7.125.3.4 insert()

Inserts a Drawable.

Parameters

in	previousElement	The element to insert the new element after.
in	d	The element to insert.

Reimplemented from Container.

7.125.3.5 remove()

Removes a Drawable. Safe to call even if drawable has not been added.

Parameters

in	d	The drawable to remove.
----	---	-------------------------

Reimplemented from Container.

7.125.3.6 removeAll()

```
void removeAll ( ) [virtual]
```

Removes all children by resetting their parent and sibling pointers. In addition, the geometry is reset and any parent is signaled of the change.

Reimplemented from Container.

7.125.3.7 setDirection()

```
void setDirection ( const\ Direction\ d ) [virtual]
```

Sets the direction of the ListLayout. If elements have already been added to the ListLayout, these elements will be repositioned to adhere to the new direction.

Parameters

d The new Direction to grow in when added children (either SOUTH or EAST).

See also

getDirection()

7.126 LockFreeDMA_Queue Class Reference

This implements a simple lock-free FIFO queue (single producer, single consumer).

#include <touchgfx/hal/DMA.hpp>

Public Member Functions

LockFreeDMA_Queue (BlitOp *mem, atomic_t n)

Constructs a lockfree queue.

• virtual bool isEmpty ()

Query if this object is empty.

· virtual bool isFull ()

Query if this object is full.

virtual void pushCopyOf (const BlitOp &op)

Push copy of the given operation.

Protected Member Functions

• virtual void pop ()

Removes the top-of-stack object.

virtual const BlitOp * first ()

Gets the first blit operation.

Protected Attributes

• BlitOp * q

Pointer to the queue memory.

· atomic_t capacity

The number of elements the queue can contain.

· atomic t head

Index to the head element.

atomic_t tail

Index to the tail element.

7.126.1 Detailed Description

This implements a simple lock-free FIFO queue (single producer, single consumer)

See also

DMA_Queue

7.126.2 Constructor & Destructor Documentation

7.126.2.1 LockFreeDMA_Queue()

```
\label{eq:lockFreeDMAQueue} \mbox{LockFreeDMA} \mbox{\@omega} \m
```

Constructs a lockfree queue.

Parameters

out	mem	Pointer to the memory used by the queue to store elements.
	n	Number of elements the memory provided can contain.

7.126.3 Member Function Documentation

```
7.126.3.1 first()
const BlitOp * first ( ) [protected], [virtual]
Returns
```

the first blitop.

Implements DMA_Queue.

```
7.126.3.2 isEmpty()
```

bool isEmpty () [virtual]
Query if this object is empty.

Returns

true if empty, false if not.

Implements DMA_Queue.

```
7.126.3.3 isFull()
```

```
bool isFull ( ) [virtual]
```

Query if this object is full.

Returns

true if full, false if not.

Implements DMA_Queue.

Push copy of the given operation.

Parameters

op The operation.

Implements DMA_Queue.

7.127 ManyBlockAllocator< block_size, blocks, bytes_pr_pixel > Class Template Reference

This class is partial framebuffer allocator using multiple blocks.

#include <touchgfx/hal/FrameBufferAllocator.hpp>

Public Member Functions

 virtual uint16_t allocateBlock (const uint16_t x, const uint16_t y, const uint16_t width, const uint16_t height, uint8_t **block)

Allocates a framebuffer block.

• virtual void markBlockReadyForTransfer ()

Marks a previously allocated block as ready to be transferred to the LCD.

virtual bool hasBlockReadyForTransfer ()

Check if a block is ready for transfer to the LCD.

virtual const uint8_t * getBlockForTransfer (Rect &rect)

Get the block ready for transfer.

• virtual void freeBlockAfterTransfer ()

Free a block after transfer to the LCD.

7.127.1 Detailed Description

```
template<uint32_t block_size, int32_t blocks, uint32_t bytes_pr_pixel> class touchgfx::ManyBlockAllocator< block_size, blocks, bytes_pr_pixel>
```

This class is partial framebuffer allocator using multiple blocks. New buffers can be allocated until no free blocks are available. After transfer to LCD, a block is queued for allocation again.

See also

SingleBlockAllocator

7.127.2 Member Function Documentation

7.127.2.1 allocateBlock()

Allocates a framebuffer block. The block will have at least the width requested. The height of the allocated block can be lower than requested if not enough memory is available.

Parameters

X	The absolute x coordinate of the block on the screen	
У	The absolute y coordinate of the block on the screen.	
width	The width of the block.	
height	The height of the block.	
block	Pointer to pointer to return the block address in.	

Returns

The height of the allocated block.

Implements FrameBufferAllocator.

7.127.2.2 freeBlockAfterTransfer()

```
void freeBlockAfterTransfer ( ) [inline], [virtual]
```

Marks a previously allocated block as transferred and ready to reuse.

 $Implements\ Frame Buffer Allocator.$

7.127.2.3 getBlockForTransfer()

Get the block ready for transfer.

Parameters

	rect	Reference to rect to write block x, y, width, and height.
۱		Therefore to reacte mile break n, j, main, and meight

Returns

Returns the address of the block ready for transfer.

Implements FrameBufferAllocator.

7.127.2.4 hasBlockReadyForTransfer()

```
bool hasBlockReadyForTransfer ( ) [inline], [virtual]
```

Check if a block is ready for transfer to the LCD.

Returns

True if a block is ready for transfer.

Implements FrameBufferAllocator.

7.127.2.5 markBlockReadyForTransfer()

```
void markBlockReadyForTransfer ( ) [inline], [virtual]
```

Marks a previously allocated block as ready to be transferred to the LCD.

Implements FrameBufferAllocator.

7.128 Matrix4x4 Class Reference

This class represents row major 4x4 homogeneous matrices.

```
#include <touchgfx/Math3D.hpp>
```

Public Member Functions

• Matrix4x4 ()

Default constructor.

· float getElement (int row, int column) const

Gets an element.

• void setViewDistance (float distance)

Sets view distance.

Matrix4x4 setElement (int row, int column, float value)

Sets an element.

Matrix4x4 & concatenateXRotation (float radians)

Concatenate x coordinate rotation.

Matrix4x4 & concatenateYRotation (float radians)

Concatenate y coordinate rotation.

Matrix4x4 & concatenateZRotation (float radians)

Concatenate z coordinate rotation.

Matrix4x4 & concatenateXTranslation (float distance)

Concatenate x coordinate translation.

Matrix4x4 & concatenateYTranslation (float distance)

Concatenate y coordinate translation.

• Matrix4x4 & concatenateZTranslation (float distance)

Concatenate z coordinate translation.

Matrix4x4 & concatenateXScale (float distance)

Concatenate x coordinate scale.

• Matrix4x4 & concatenateYScale (float distance)

Concatenate y coordinate scale.

• Matrix4x4 & concatenateZScale (float distance)

Concatenate z coordinate scale.

Static Public Member Functions

• static Matrix4x4 identity ()

Gets the identity.

Protected Attributes

float elements [4][4]
 The elements [4][4].

7.128.1 Detailed Description

This class represents row major 4x4 homogeneous matrices.

7.128.2 Constructor & Destructor Documentation

```
7.128.2.1 Matrix4x4()
```

```
Matrix4x4 ( )
```

Default constructor.

7.128.3 Member Function Documentation

7.128.3.1 concatenateXRotation()

Concatenate x coordinate rotation.

Parameters

radians	The radians.
radians	Tric radians.

Returns

A matrix_4x4&

7.128.3.2 concatenateXScale()

Concatenate x coordinate scale.

Parameters

```
distance The distance.
```

Returns

A matrix_4x4&

7.128.3.3 concatenateXTranslation()

Concatenate x coordinate translation.

Parameters

Returns

A matrix_4x4&

7.128.3.4 concatenateYRotation()

Concatenate y coordinate rotation.

Parameters

```
radians The radians.
```

Returns

A matrix_4x4&

7.128.3.5 concatenateYScale()

Concatenate y coordinate scale.

Parameters

distance	The distance.
----------	---------------

Returns

A matrix_4x4&

7.128.3.6 concatenateYTranslation()

Concatenate y coordinate translation.

Parameters

distance	The distance.
aiotarioc	THE distance.

Returns

A matrix_4x4&

7.128.3.7 concatenateZRotation()

```
\begin{tabular}{lll} Matrix 4x4 & concatenate ZR otation ( \\ & float \ radians ) \end{tabular}
```

Concatenate z coordinate rotation.

Parameters

radians The radians	3.
---------------------	----

Returns

A matrix_4x4&

7.128.3.8 concatenateZScale()

Concatenate z coordinate scale.

Parameters

distance	The distance.
uistance	i i ne distance.

Returns

A matrix_4x4&

7.128.3.9 concatenateZTranslation()

Concatenate z coordinate translation.

Parameters

distance	The distance.
----------	---------------

Returns

A matrix_4x4&

7.128.3.10 getElement()

```
float getElement (
          int row,
          int column ) const [inline]
```

Gets an element.

Parameters

row	The row.
column	The column.

Returns

The element.

7.128.3.11 identity()

```
static Matrix4x4 identity ( ) [inline], [static]
```

Gets the identity. Instead of using "Matrix4x4::identity()" consider using "Matrix4x4()" instead.

Returns

A matrix_4x4.

7.128.3.12 setElement()

```
int column,
float value ) [inline]
```

Sets an element.

Parameters

row	The row.
column	The column.
value	The value.

Returns

A matrix_4x4&

7.128.3.13 setViewDistance()

Sets view distance.

Parameters

distance	The distance.
----------	---------------

7.129 MCUInstrumentation Class Reference

Interface for instrumenting processors to measure MCU load via measured CPU cycles.

#include <platform/core/MCUInstrumentation.hpp>

Public Member Functions

• MCUInstrumentation ()

Constructor.

• virtual void init ()=0

Initialize.

virtual ~MCUInstrumentation ()

Virtual destructor.

- virtual unsigned int getElapsedUS (unsigned int start, unsigned int now, unsigned int clockfrequency)=0
 Gets elapsed microseconds based on clock frequency.
- virtual unsigned int getCPUCycles (void)=0

Gets CPU cycles from register.

• virtual void setMCUActive (bool active)

Sets MCU activity high.

virtual uint32_t getCCConsumed ()

Gets number of consumed clock cycles.

virtual void setCCConsumed (uint32_t val)

Sets number of consumed clock cycles.

Protected Attributes

- volatile uint32_t cc_consumed
 Amount of consumed CPU cycles.
- volatile uint32_t cc_in
 Current CPU cycles.

7.129.1 Detailed Description

Interface for instrumenting processors to measure MCU load via measured CPU cycles.

7.129.2 Constructor & Destructor Documentation

```
7.129.2.1 MCUInstrumentation()

MCUInstrumentation ( ) [inline]

Constructor. Initializes members.

7.129.2.2 ~MCUInstrumentation()

~MCUInstrumentation ( ) [inline], [virtual]
```

7.129.3 Member Function Documentation

Virtual destructor.

Gets CPU cycles from register.

CPU cycles.

Returns

7.129.3.3 getElapsedUS()

```
unsigned int getElapsedUS (
         unsigned int start,
         unsigned int now,
         unsigned int clockfrequency) [pure virtual]
```

Gets elapsed microseconds based on clock frequency.

Parameters

start	Start time.
now	Current time.
clockfrequency	Clock frequency of the system expressed in MHz.

Returns

Elapsed microseconds start and now.

7.129.3.4 init()

```
void init ( ) [pure virtual]
Initialize.
```

7.129.3.5 setCCConsumed()

```
void setCCConsumed (
          uint32_t val ) [inline], [virtual]
```

Sets number of consumed clock cycles.

Parameters

val	number of clock cycles.
-----	-------------------------

7.129.3.6 setMCUActive()

```
void setMCUActive (
                bool active ) [inline], [virtual]
```

Sets MCU activity high.

Parameters

7.130 ModalWindow Class Reference

Container for displaying a modal window and hijacking touch event to underlaying view and widgets.

#include <include/gui/common/ModalWindow.hpp>

Public Member Functions

• ModalWindow ()

Default constructor.

virtual ∼ModalWindow ()

Destructor.

virtual void setBackground (const Bitmapld &bmpld)

Sets the background of the actual window.

virtual void setBackground (const BitmapId &bmpId, int16_t backgroundX, int16_t backgroundY)

Sets the background of the actual window.

virtual uint16_t getBackgroundWidth () const

Gets the width of the actual window (the background images).

• virtual uint16_t getBackgroundHeight () const

Gets the height of the actual window (the background images).

virtual void add (Drawable &d)

Adds a drawable to the ModalWindow.

• virtual void remove (Drawable &d)

Removes the drawable from the ModalWindow.

virtual void setShadeAlpha (uint8_t alpha)

Sets the alpha value of the background shade.

• virtual uint8_t getShadeAlpha () const

Gets the alpha value of the background shade.

• virtual void setShadeColor (colortype color)

Sets the color of the background shade.

· virtual colortype getShadeColor () const

Gets the color of the background shade.

· virtual void show ()

Make the ModalWindow visible.

· virtual void hide ()

Make the ModalWindow invisible.

· virtual bool isShowing () const

Query if this ModalWindow is showing.

Protected Attributes

· Box backgroundShade

The background shade.

· Container windowContainer

The window container that defines the active container area where both the windowBackground and added drawables are placed.

· Image windowBackground

The window background.

Additional Inherited Members

7.130.1 Detailed Description

Container for displaying a modal window and hijacking touch event to underlaying view and widgets. The container has a background image and a surround box that acts as a shade on top of the rest of the screen. The background image must be set (using the setBackground method) and the shade can be adjusted (using the setShadeAlpha and setShadeColor methods).

The ModalWindow can either be used directly by adding widgets/containers to the ModalWindow from your view or by sub-classing it if you need a specific ModalWindow with predefined behavior across your application.

The ModalWindow should be instantiated in the view class and added as the last element (to always be on top). The ModalWindow will fill up the entire screen so it should always be placed at x=0, y=0.

To control the visibility of the ModalWindow use the show and hide methods.

7.130.2 Constructor & Destructor Documentation

```
7.130.2.1 ModalWindow()
```

```
ModalWindow ( )
```

Default constructor.

```
7.130.2.2 \sim ModalWindow()
```

```
~ModalWindow ( ) [virtual]
```

Destructor.

7.130.3 Member Function Documentation

```
7.130.3.1 add()  \mbox{void add (} \\ \mbox{Drawable & $d$ ) [virtual]}
```

Adds a drawable to the ModalWindow. The drawable will be placed relative to the background image.

Parameters

```
in d The drawable to add.
```

Reimplemented from Container.

7.130.3.2 getBackgroundHeight()

```
uint16_t getBackgroundHeight ( ) const [virtual]
```

Gets the height of the actual window (the background images). Whereas the getHeight() method will return the

height including the shade.

Returns

The height of the actual window.

7.130.3.3 getBackgroundWidth()

```
uint16_t getBackgroundWidth ( ) const [virtual]
```

Gets the width of the actual window (the background images). Whereas the <code>getWidth()</code> method will return the width including the shade.

Returns

The width of the actual window.

7.130.3.4 getShadeAlpha()

```
uint8_t getShadeAlpha ( ) const [virtual]
```

Gets the alpha value of the background shade.

Returns

The background shades alpha.

7.130.3.5 getShadeColor()

```
colortype getShadeColor ( ) const [virtual]
```

Gets the color of the background shade.

Returns

The color of the background shade.

7.130.3.6 hide()

```
void hide ( ) [virtual]
```

Make the ModalWindow invisible.

7.130.3.7 isShowing()

```
bool isShowing ( ) const [virtual]
```

Query if this ModalWindow is showing.

Returns

true if showing, false if not.

7.130.3.8 remove()

Removes the drawable from the ModalWindow.

Parameters

```
in d The drawable to remove.
```

Reimplemented from Container.

Sets the background of the actual window. The remaining area of the screen will be covered by the shade. The background image is centered on the screen.

Parameters

bmp⇔	Identifier for the background bitmap.	
ld		

7.130.3.10 setBackground() [2/2]

Sets the background of the actual window. The remaining area of the screen will be covered by the shade. The background image will be placed at the backgroundX and backgroundY coordinate.

Parameters

bmpld	Identifier for the bitmap.
backgroundX	The background x coordinate.
backgroundY	The background y coordinate.

7.130.3.11 setShadeAlpha()

Sets the alpha value of the background shade. Default = 96.

Parameters

alpha	The new alpha.

7.130.3.12 setShadeColor()

Sets the color of the background shade. Default = Black.

Parameters

```
color The new color.
```

7.130.3.13 show()

```
void show ( ) [virtual]
```

Make the ModalWindow visible.

7.131 MoveAnimator < T > Class Template Reference

A MoveAnimator makes the template class T able to animate a movement.

```
#include <touchgfx/mixins/MoveAnimator.hpp>
```

Public Member Functions

• MoveAnimator ()

Default constructor.

virtual ∼MoveAnimator ()

Destructor.

void setMoveAnimationEndedAction (GenericCallback< const MoveAnimator< T > & > &callback)

Associates an action to be performed when the animation ends.

void clearMoveAnimationEndedAction ()

Clears the move animation ended action previously set by setMoveAnimationEndedAction.

virtual void setMoveAnimationDelay (uint16_t delay)

Sets a delay on animations done by the MoveAnimator.

virtual uint16_t getMoveAnimationDelay () const

Gets the current animation delay.

virtual bool isRunning () const

Gets whether or not the move animation is running.

virtual bool isMoveAnimationRunning () const

Gets whether or not the move animation is running.

void startMoveAnimation (int16_t endX, int16_t endY, uint16_t duration, EasingEquation xProgression
 Equation=&EasingEquations::linearEaseNone, EasingEquation yProgressionEquation=&EasingEquations
 ::linearEaseNone)

Starts the move animation.

• void cancelMoveAnimation ()

Cancel move animation.

Protected Member Functions

virtual void handleTickEvent ()

The tick handler that handles the actual animation steps.

void nextMoveAnimationStep ()

Execute next step in move animation.

Protected Attributes

· bool moveAnimationRunning

Boolean that is true if the animation is running.

· uint16_t moveAnimationCounter

Counter that is equal to the current step in the animation.

uint16_t moveAnimationDelay

A delay that is applied before animation start. Expressed in ticks.

· uint16 t moveAnimationDuration

The complete duration of the animation. Expressed in ticks.

· int16 t moveAnimationStartX

The X value at the beginning of the animation.

int16_t moveAnimationStartY

The Y value at the beginning of the animation.

int16 t moveAnimationEndX

The X value at the end of the animation.

int16 t moveAnimationEndY

The Y value at the end of the animation.

• EasingEquation moveAnimationXEquation

EasingEquation expressing the development of the X value during the animation.

• EasingEquation moveAnimationYEquation

EasingEquation expressing the development of the Y value during the animation.

GenericCallback < const MoveAnimator < T > &> * moveAnimationEndedCallback

Animation ended Callback.

7.131.1 Detailed Description

$$\label{template} \begin{split} \text{template} &< \text{class T}> \\ \text{class touchgfx::} &\text{MoveAnimator} < \text{T}> \end{split}$$

A MoveAnimator makes the template class T able to animate a movement from its current position to a specified end position. The movement in both the X and Y direction can be described by supplying EasingEquations. The MoveAnimator performs a callback when the animation has finished.

This mixin can be used on any Drawable.

Template Parameters

T | Specifies the type should have the move animation capability.

7.131.2 Constructor & Destructor Documentation

7.131.2.1 MoveAnimator()

```
MoveAnimator ( ) [inline]
```

Default constructor. Creates and initialize the MoveAnimator.

7.131.2.2 \sim MoveAnimator()

```
~MoveAnimator ( ) [inline], [virtual]
```

Destructor. Destroys the MoveAnimator.

7.131.3 Member Function Documentation

7.131.3.1 cancelMoveAnimation()

```
void cancelMoveAnimation ( ) [inline]
```

Cancel move animation.

7.131.3.2 clearMoveAnimationEndedAction()

```
void clearMoveAnimationEndedAction ( ) [inline]
```

Clears the move animation ended action previously set by setMoveAnimationEndedAction.

See also

setMoveAnimationEndedAction

7.131.3.3 getMoveAnimationDelay()

```
uint16_t getMoveAnimationDelay ( ) const [inline], [virtual]
```

Gets the current animation delay.

Returns

The current animation delay.

7.131.3.4 handleTickEvent()

```
void handleTickEvent ( ) [inline], [protected], [virtual]
```

The tick handler that handles the actual animation steps.

7.131.3.5 isMoveAnimationRunning()

```
bool isMoveAnimationRunning ( ) const [inline], [virtual]
```

Gets whether or not the move animation is running.

Returns

true if the move animation is running.

7.131.3.6 isRunning()

```
bool isRunning ( ) const [inline], [virtual]
```

Gets whether or not the move animation is running.

Returns

true if the move animation is running.

7.131.3.7 nextMoveAnimationStep()

```
void nextMoveAnimationStep ( ) [inline], [protected]
```

Execute next step in move animation and stop the timer if necessary.

7.131.3.8 setMoveAnimationDelay()

Sets a delay on animations done by the MoveAnimator.

Parameters

delay The delay in ticks.

7.131.3.9 setMoveAnimationEndedAction()

Associates an action to be performed when the animation ends.

Parameters

callback The callback to be executed. The callback will be given a reference to the MoveAnimator.

See also

GenericCallback

7.131.3.10 startMoveAnimation()

```
void startMoveAnimation (
```

```
int16_t endX,
int16_t endY,
uint16_t duration,
EasingEquation xProgressionEquation = &EasingEquations::linearEaseNone,
EasingEquation yProgressionEquation = &EasingEquations::linearEaseNone ) [inline]
```

Starts the move animation from the current position to the specified end position. The development of the position (X, Y) during the animation is described by the supplied EasingEquations.

Parameters

endX	The X position of T at animation end. Relative to the container or view that holds T.
endY	The Y position of T at animation end. Relative to the container or view that holds T.
duration	The duration of the animation measured in ticks.
xProgressionEquation	The equation that describes the development of the X position during the animation. Default = EasingEquations::linearEaseNone.
yProgressionEquation	The equation that describes the development of the Y position during the animation. Default = EasingEquations::linearEaseNone.

7.132 MVPApplication Class Reference

A specialization of the TouchGFX Application class.

```
#include <mvp\MVPApplication.hpp>
```

Public Member Functions

MVPApplication ()

Default constructor.

virtual ∼MVPApplication ()

Destructor.

• virtual void handlePendingScreenTransition ()

Handles the pending screen transition.

Protected Member Functions

void evaluatePendingScreenTransition ()
 Evaluates the pending Callback instances.

Protected Attributes

• Presenter * currentPresenter

Pointer to the currently active presenter.

GenericCallback * pendingScreenTransitionCallback

Callback for screen transitions. Will be set to something valid when a transition request is made.

Additional Inherited Members

7.132.1 Detailed Description

A specialization of the TouchGFX Application class that provides the necessary glue for transitioning between presenter/view pairs.

It maintains a callback for transitioning and evaluates this at each tick.

See also

Application

7.132.2 Constructor & Destructor Documentation

```
7.132.2.1 MVPApplication()
```

MVPApplication () [inline]

Default constructor.

```
7.132.2.2 \simMVPApplication()
```

```
\sim\!\!\text{MVPApplication} ( ) [inline], [virtual]
```

Destructor.

7.132.3 Member Function Documentation

7.132.3.1 evaluatePendingScreenTransition()

```
void evaluatePendingScreenTransition ( ) [inline], [protected]
```

Evaluates the pending Callback instances. If a callback is valid, it is executed and a Screen transition is executed.

7.132.3.2 handlePendingScreenTransition()

```
void handlePendingScreenTransition ( ) [inline], [virtual]
```

Delegates the work to evaluatePendingScreenTransition()

Reimplemented from Application.

7.133 MVPHeap Class Reference

Generic heap class for MVP applications.

```
#include <mvp/MVPHeap.hpp>
```

Public Member Functions

- MVPHeap (AbstractPartition &pres, AbstractPartition &scr, AbstractPartition &tra, MVPApplication &app)
 Constructor.
- virtual ∼MVPHeap ()

Destructor.

Public Attributes

AbstractPartition & presenterStorage

A memory partition containing enough memory to hold the largest presenter.

AbstractPartition & screenStorage

A memory partition containing enough memory to hold the largest view.

· AbstractPartition & transitionStorage

A memory partition containing enough memory to hold the largest transition.

MVPApplication & frontendApplication

A reference to the MVPApplication instance.

7.133.1 Detailed Description

Generic heap class for MVP applications. Serves as a way of obtaining the memory storage areas for presenters, screens, transitions and the concrete application.

Subclassed by an application-specific heap which provides the actual storage areas. This generic interface is used only in makeTransition.

7.133.2 Constructor & Destructor Documentation

7.133.2.1 MVPHeap()

```
MVPHeap (

AbstractPartition & pres,

AbstractPartition & scr,

AbstractPartition & tra,

MVPApplication & app ) [inline]
```

Constructs an MVPHeap.

Parameters

in	pres	A memory partition containing enough memory to hold the largest presenter.
in	scr	A memory partition containing enough memory to hold the largest view.
in	tra	A memory partition containing enough memory to hold the largest transition.
in	арр	A reference to the MVPApplication instance.

```
7.133.2.2 ∼MVPHeap()
```

```
~MVPHeap ( ) [inline], [virtual] Destructor.
```

7.134 NoDMA Class Reference

This is an "empty" DMA subclass that does nothing except assert if accidentally used.

```
#include <touchgfx/hal/NoDMA.hpp>
```

Public Member Functions

• NoDMA ()

Default constructor.

• virtual BlitOperations getBlitCaps ()

No blit operations supported by this DMA implementation.

virtual void setupDataCopy (const BlitOp &blitOp)

Asserts if used.

virtual void setupDataFill (const BlitOp &blitOp)

Asserts if used.

• virtual void signalDMAInterrupt ()

Does nothing.

· virtual void flush ()

Does nothing.

Additional Inherited Members

7.134.1 Detailed Description

This is an "empty" DMA subclass that does nothing except assert if accidentally used. An instance of this object can be used if DMA support is not desired.

See also

DMA_Interface

7.134.2 Constructor & Destructor Documentation

```
7.134.2.1 NoDMA()
```

```
NoDMA ( ) [inline]
```

Constructs a NoDMA object, with a queue of 1 element.

7.134.3 Member Function Documentation

```
7.134.3.1 flush()
```

```
void flush ( ) [inline], [virtual]
```

Block until all DMA transfers are complete. Since this particular DMA does not do anything, return immediately. Reimplemented from DMA_Interface.

```
7.134.3.2 getBlitCaps()
```

```
BlitOperations getBlitCaps ( ) [inline], [virtual]
```

No blit operations supported by this DMA implementation.

Returns

Zero (no blit ops supported).

Implements DMA_Interface.

7.134.3.3 setupDataCopy()

Asserts if used.

Parameters

blitOp The blit operation to be performed by this DMA instance.

Implements DMA Interface.

7.134.3.4 setupDataFill()

Asserts if used.

Parameters

blitOp The blit operation to be performed by this DMA instance.

Implements DMA_Interface.

7.134.3.5 signalDMAInterrupt()

```
void signalDMAInterrupt ( ) [inline], [virtual]
```

Does nothing.

Implements DMA_Interface.

7.135 NoTouchController Class Reference

#include <NoTouchController.hpp>

Public Member Functions

· virtual void init ()

Empty initialization.

virtual ∼NoTouchController ()

Destructor.

virtual bool sampleTouch (int32_t &x, int32_t &y)

Does nothing.

7.135.1 Detailed Description

Empty TouchController implementation which does nothing. Use this if your display does not have touch input capabilities.

7.135.2 Constructor & Destructor Documentation

```
7.135.2.1 ~NoTouchController()
```

```
\simNoTouchController ( ) [inline], [virtual]
```

Destructor.

7.135.3 Member Function Documentation

```
7.135.3.1 init()
```

```
void init ( ) [inline], [virtual]
```

Empty initialization.

Implements TouchController.

7.135.3.2 sampleTouch()

Parameters

out	Х	unused.
out	У	unused.

Returns

false.

Implements TouchController.

7.136 NoTransition Class Reference

The most simple Transition without any visual effects.

```
#include <touchgfx/transitions/NoTransition.hpp>
```

Public Member Functions

• NoTransition ()

Default constructor.

virtual ∼NoTransition ()

Destructor.

• virtual void handleTickEvent ()

Indicates that the transition is done after the first tick.

Additional Inherited Members

7.136.1 Detailed Description

The most simple Transition without any visual effects.

See also

Transition

7.136.2 Constructor & Destructor Documentation

```
7.136.2.1 NoTransition()
```

```
NoTransition ( ) [inline]
```

Default constructor.

7.136.2.2 \sim NoTransition()

```
\sim NoTransition ( ) [inline], [virtual]
```

Destructor.

7.136.3 Member Function Documentation

```
7.136.3.1 handleTickEvent()
```

```
void handleTickEvent ( ) [inline], [virtual]
```

Indicates that the transition is done after the first tick.

Reimplemented from Transition.

7.137 OSWrappers Class Reference

This class specifies OS wrappers for dealing with the frame buffer semaphore and the VSYNC signal.

```
#include <touchgfx/hal/OSWrappers.hpp>
```

Static Public Member Functions

• static void initialize ()

Initialize frame buffer semaphore and queue/mutex for VSYNC signal.

static void signalVSync ()

Signal that a VSYNC has occurred.

• static void waitForVSync ()

This function blocks until a VSYNC occurs.

• static void takeFrameBufferSemaphore ()

Take the frame buffer semaphore.

• static void tryTakeFrameBufferSemaphore ()

Attempt to obtain the frame buffer semaphore.

• static void giveFrameBufferSemaphore ()

Release the frame buffer semaphore.

static void giveFrameBufferSemaphoreFromISR ()

Release the frame buffer semaphore in a way that is safe in interrupt context. Called from ISR.

static void taskDelay (uint16_t ms)

A function that causes executing task to sleep for a number of milliseconds.

7.137.1 Detailed Description

This class specifies OS wrappers for dealing with the frame buffer semaphore and the VSYNC signal.

7.137.2 Member Function Documentation

7.137.2.1 giveFrameBufferSemaphore()

```
static void giveFrameBufferSemaphore ( ) [static]
```

Release the frame buffer semaphore.

7.137.2.2 giveFrameBufferSemaphoreFromISR()

```
\verb|static| void giveFrameBufferSemaphoreFromISR () [static]|\\
```

Release the frame buffer semaphore in a way that is safe in interrupt context. Called from ISR.

7.137.2.3 initialize()

```
static void initialize ( ) [static]
```

Initialize frame buffer semaphore and queue/mutex for VSYNC signal.

7.137.2.4 signalVSync()

```
static void signalVSync ( ) [static]
```

Signal that a VSYNC has occurred. Should make the vsync queue/mutex available.

Note

This function is called from an ISR, and should (depending on OS) trigger a scheduling.

7.137.2.5 takeFrameBufferSemaphore()

```
static void takeFrameBufferSemaphore ( ) [static]
```

Take the frame buffer semaphore. Blocks until semaphore is available.

7.137.2.6 taskDelay()

A function that causes executing task to sleep for a number of milliseconds. This function is OPTIONAL. It is only used by the TouchGFX in the case of a specific frame refresh strategy (REFRESH_STRATEGY_OPTIM_SING LE_BUFFER_TFT_CTRL). Due to backwards compatibility, in order for this function to be useable by the HAL the function must be explicitly registered: hal.registerTaskDelayFunction(&OSWrappers::taskDelay)

Parameters

```
ms The number of milliseconds to sleep
```

See also

HAL::setFrameRefreshStrategy(FrameRefreshStrategy s) HAL::registerTaskDelayFunction(void (*delayF)(uint16_t))

7.137.2.7 tryTakeFrameBufferSemaphore()

```
static void tryTakeFrameBufferSemaphore ( ) [static]
```

Attempt to obtain the frame buffer semaphore. If semaphore is not available, do nothing.

Note

must return immediately! This function does not care who has the taken the semaphore, it only serves to make sure that the semaphore is taken by someone.

7.137.2.8 waitForVSync()

```
static void waitForVSync ( ) [static]
```

This function blocks until a VSYNC occurs.

Note

This function must first clear the mutex/queue and then wait for the next one to occur.

7.138 Outline Class Reference

An internal class that implements the main rasterization algorithm.

```
#include <touchgfx/canvas_widget_renderer/Outline.hpp>
```

Public Types

· typedef unsigned int OutlineFlags_t

Defines an alias representing the outline flags.

Public Member Functions

• Outline ()

Default constructor.

virtual ∼Outline ()

Destructor.

· void reset ()

Resets this object.

void moveTo (int x, int y)

Move a virtual pen to the specified coordinate.

void lineTo (int x, int y)

Create a line from the current virtual pen coordinate to the given coordinate creating an Outline.

• unsigned getNumCells () const

Gets number cells registered in the current drawn path for the Outline.

const Cell * getCells ()

Gets a pointer to the the Cell objects in the Outline.

void setMaxRenderY (int y)

Sets maximum render y coordinate.

bool wasOutlineTooComplex ()

Determines if there was enough memory to register the entire outline.

Static Public Attributes

• static const OutlineFlags t OUTLINE NOT CLOSED = 1U

If this bit is set in flags, the current Outline has not yet been closed. Used for automatic closing an Outline before rendering the Outline.

• static const OutlineFlags_t OUTLINE_SORT_REQUIRED = 2U

If this bit is set in flags, Cell objects have been added to the Outline requiring the Cell list needs to be sorted.

7.138.1 Detailed Description

An internal class that implements the main rasterization algorithm. Used in the Rasterizer. Should not be used directly.

7.138.2 Member Typedef Documentation

```
7.138.2.1 OutlineFlags_t
unsigned int OutlineFlags_t
```

Defines an alias representing the outline flags.

7.138.3 Constructor & Destructor Documentation

```
7.138.3.1 Outline()
Outline ( )
Default constructor.
7.138.3.2 ~Outline()
~Outline ( ) [virtual]
```

7.138.4 Member Function Documentation

```
7.138.4.1 getCells()
const Cell * getCells ( )
```

Gets a pointer to the the Cell objects in the Outline. If the Outline is not closed, it is closed. If the Outline is unsorted, it will be quick sorted first.

Returns

Destructor.

A pointer to the sorted list of Cell objects in the Outline.

```
7.138.4.2 getNumCells()
unsigned getNumCells ( ) const [inline]
```

Gets number cells registered in the current drawn path for the Outline.

Returns

The number of cells.

7.138.4.3 lineTo()

Create a line from the current virtual pen coordinate to the given coordinate creating an Outline.

Parameters

X	The x coordinate.
У	The y coordinate.

7.138.4.4 moveTo()

```
void moveTo (  \mbox{int } x, \\ \mbox{int } y \mbox{)}
```

Move a virtual pen to the specified coordinate.

Parameters

X	The x coordinate.
У	The y coordinate.

7.138.4.5 reset()

```
void reset ( )
```

Resets this object. This implies removing the current Cell objects and preparing for a new Outline.

7.138.4.6 setMaxRenderY()

```
\begin{tabular}{ll} \beg
```

Sets maximum render y coordinate. This is used to avoid registering any Cell that has a y coordinate less than zero of higher than the given y.

Parameters

```
y The max y coordinate to render for the Outline.
```

7.138.4.7 wasOutlineTooComplex()

```
bool wasOutlineTooComplex ( ) [inline]
```

Determines if there was enough memory to register the entire outline, of if the outline was too complex.

Returns

false if the buffer for Outline Cell objects was too small.

7.139 PainterABGR2222 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/PainterABGR2222.hpp>

Public Member Functions

• PainterABGR2222 (colortype color=0, uint8_t alpha=255)

Constructor.

void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8 t painterColor

The color.

uint8_t painterRed

The red part of the color, scaled up to [0..255].

· uint8 t painterGreen

The green part of the color, scaled up to [0..255].

uint8_t painterBlue

The blue part of the color, scaled up to [0..255].

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.139.1 Detailed Description

The PainterABGR2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.139.2 Constructor & Destructor Documentation

7.139.2.1 PainterABGR2222()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.139.3 Member Function Documentation

```
7.139.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.139.3.2 getColor()

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.139.3.3 render()

```
int y,
unsigned count,
const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterABGR2222.

7.139.3.4 renderNext()

```
virtual bool renderNext (
            uint8_t & red,
            uint8_t & green,
            uint8_t & blue,
            uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterABGR2222.

7.139.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.	

7.139.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.140 PainterABGR2222Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterABGR2222Bitmap.hpp>

Public Member Functions

• PainterABGR2222Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapABGR2222Pointer

Pointer to the bitmap (ABGR2222)

· Bitmap bitmap

The bitmap to be used when painting.

• Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.140.1 Detailed Description

PainterABGR2222Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.140.2 Constructor & Destructor Documentation

7.140.2.1 PainterABGR2222Bitmap()

```
PainterABGR2222Bitmap (
            const Bitmap & bmp = Bitmap(BITMAP_INVALID),
            uint8_t alpha = 255 )
```

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.140.3 Member Function Documentation

```
7.140.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.140.3.2 render()

```
int xAdjust,
int y,
unsigned count,
const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterABGR2222.

7.140.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterABGR2222.

7.140.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.	
out	green	The green.	
out	blue	The blue.	
out	alpha	The alpha.	

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterABGR2222.

7.140.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.
-------	--

7.140.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp The bitmap.

7.141 PainterARGB2222 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/PainterARGB2222.hpp>
```

Public Member Functions

• PainterARGB2222 (colortype color=0, uint8_t alpha=255)

Constructor

• void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

· colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterColor

The color.

· uint8 t painterRed

The red part of the color, scaled up to [0..255].

uint8_t painterGreen

The green part of the color, scaled up to [0..255].

uint8_t painterBlue

The blue part of the color, scaled up to [0..255].

uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.141.1 Detailed Description

The PainterARGB2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.141.2 Constructor & Destructor Documentation

7.141.2.1 PainterARGB2222()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.141.3 Member Function Documentation

```
7.141.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

```
7.141.3.2 getColor()
```

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.141.3.3 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterARGB2222.

7.141.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterARGB2222.

7.141.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha T	he alpha value to use.
---------	------------------------

7.141.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.142 PainterARGB2222Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterARGB2222Bitmap.hpp>
```

Public Member Functions

- PainterARGB2222Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 Constructor.
- void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapARGB2222Pointer

Pointer to the bitmap (ARGB2222)

Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.142.1 Detailed Description

PainterARGB2222Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.142.2 Constructor & Destructor Documentation

7.142.2.1 PainterARGB2222Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.142.3 Member Function Documentation

```
7.142.3.1 getAlpha()
uint8_t getAlpha ( ) const
Gets the current alpha value.
```

Returns

The current alpha value.

See also

setAlpha

7.142.3.2 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Χ	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterARGB2222.

```
7.142.3.3 renderInit()
```

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterARGB2222.

7.142.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterARGB2222.

7.142.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.
-------	--

7.142.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp	The bitmap.	

7.143 PainterARGB8888 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/PainterARGB8888.hpp>

Public Member Functions

• PainterARGB8888 (colortype color=0, uint8 t alpha=255)

Constructor.

• void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

· colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

uint8 t painterRed

The red part of the color.

• uint8_t painterGreen

The green part of the color.

• uint8_t painterBlue

The blue part of the color.

• uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.143.1 Detailed Description

The PainterARGB8888 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.143.2 Constructor & Destructor Documentation

7.143.2.1 PainterARGB8888()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.143.3 Member Function Documentation

```
7.143.3.1 getAlpha()
uint8_t getAlpha ( ) const
Gets the current alpha value.
```

Returns

The current alpha value.

See also

setAlpha

```
7.143.3.2 getColor()
```

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.143.3.3 render()

```
virtual void render (
        uint8_t * ptr,
        int x,
        int xAdjust,
        int y,
        unsigned count,
        const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterARGB8888.

7.143.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterARGB8888.

7.143.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

	The alpha value to use.
l ainha l	The alpha value to use
aipiia	The dipha value to doe.

7.143.3.6 setColor()

```
void setColor (
```

```
colortype color,
uint8_t alpha = 255 )
```

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.144 PainterARGB8888Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterARGB8888Bitmap.hpp>

Public Member Functions

• PainterARGB8888Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

· uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint32 t * bitmapARGB8888Pointer

Pointer to the bitmap (ARGB8888)

const uint16_t * bitmapRGB565Pointer

Pointer to the bitmap (RGB565)

• const uint8_t * bitmapRGB888Pointer

Pointer to the bitmap (RGB888)

· Bitmap bitmap

The bitmap to be used when painting.

• Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use combined with alpha data from the bitmap.

Additional Inherited Members

7.144.1 Detailed Description

PainterARGB8888Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.144.2 Constructor & Destructor Documentation

7.144.2.1 PainterARGB8888Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.144.3 Member Function Documentation

```
7.144.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.144.3.2 render()

```
int xAdjust,
int y,
unsigned count,
const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	x The x coordinate.		
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterARGB8888.

7.144.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterARGB8888.

7.144.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterARGB8888.

7.144.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

7.144.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp	The bitmap.
-----	-------------

7.145 PainterARGB8888L8Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterARGB8888L8Bitmap.hpp>
```

Public Member Functions

- PainterARGB8888L8Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 - Constructor.
- void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapPointer

Pointer to the bitmap (L8)

const uint8 t * bitmapExtraPointer

Pointer to the CLUT (L8)

· Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use combined with alpha data from the bitmap.

Additional Inherited Members

7.145.1 Detailed Description

PainterARGB8888L8Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.145.2 Constructor & Destructor Documentation

7.145.2.1 PainterARGB8888L8Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.145.3 Member Function Documentation

```
7.145.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.145.3.2 render()

```
virtual void render (
        uint8_t * ptr,
        int x,
        int xAdjust,
        int y,
        unsigned count,
        const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	X	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterARGB8888.

7.145.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterARGB8888.

7.145.3.4 renderNext()

```
virtual bool renderNext (
            uint8_t & red,
            uint8_t & green,
```

```
uint8_t & blue,
uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterARGB8888.

7.145.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.

7.145.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

```
bmp The bitmap.
```

7.146 PainterBGRA2222 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/PainterBGRA2222.hpp>
```

Public Member Functions

PainterBGRA2222 (colortype color=0, uint8_t alpha=255)

Constructor.

• void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterColor

The color.

· uint8 t painterRed

The red part of the color, scaled up to [0..255].

uint8_t painterGreen

The green part of the color, scaled up to [0..255].

• uint8_t painterBlue

The blue part of the color, scaled up to [0..255].

uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.146.1 Detailed Description

The PainterBGRA2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.146.2 Constructor & Destructor Documentation

7.146.2.1 PainterBGRA2222()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.146.3 Member Function Documentation

```
7.146.3.1 getAlpha()
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

```
7.146.3.2 getColor()
```

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.146.3.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	X	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	

Parameters

У	The y coordinate.
count	Number of pixels to fill.
covers	The coverage in of each pixel.

Reimplemented from AbstractPainterBGRA2222.

7.146.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterBGRA2222.

7.146.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.
-------	-------------------------

7.146.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.147 PainterBGRA2222Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterBGRA2222Bitmap.hpp>

Public Member Functions

• PainterBGRA2222Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8 t * bitmapBGRA2222Pointer

Pointer to the bitmap (BGRA2222)

Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.147.1 Detailed Description

PainterBGRA2222Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.147.2 Constructor & Destructor Documentation

7.147.2.1 PainterBGRA2222Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.147.3 Member Function Documentation

```
7.147.3.1 getAlpha()
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.147.3.2 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterBGRA2222.

7.147.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterBGRA2222.

7.147.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterBGRA2222.

7.147.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha The alpha value to use if there is no alpha channel in the bitmap.

7.147.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

```
bmp The bitmap.
```

7.148 PainterBW Class Reference

A Painter that will paint using a color on a LCD1bpp display.

```
#include <touchgfx/widgets/canvas/PainterBW.hpp>
```

Public Member Functions

• void setColor (colortype color)

Sets color to use when drawing the CanvasWidget.

colortype getColor () const

Gets the current color.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Static Public Member Functions

• static unsigned bw (unsigned red, unsigned green, unsigned blue)

Convert color to black/white.

Protected Member Functions

virtual bool renderNext (uint8_t &color)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterColor

The color to use when painting.

Additional Inherited Members

7.148.1 Detailed Description

PainterBW is used for drawing one 1bpp displays. The color is either on or off No transparency is supported.

See also

AbstractPainter

7.148.2 Member Function Documentation

7.148.2.1 bw()

```
static unsigned bw (
          unsigned red,
          unsigned green,
          unsigned blue ) [static]
```

Converts the selected color to either white (1) or black (0) depending on the converted gray value.

Parameters

red	The red color.		
green	The green color.		
blue	The blue color.		

Returns

1 (white) if the brightness of the RGB color is above 50% and 0 (black) otherwise.

7.148.2.2 getColor()

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.148.2.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	X	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterBW.

7.148.2.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	color	Color of the pixel, 0 or 1.
-----	-------	-----------------------------

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterBW.

7.148.2.5 setColor()

Sets color to use when drawing the CanvasWidget.

Parameters

color The color, 0=black, otherwise white.

7.149 PainterBWBitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterBWBitmap.hpp>

Public Member Functions

• PainterBWBitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID))

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &color)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapBWPointer

Pointer to the bitmap (BW)

• LCD1bpp::bwRLEdata bw rle

Pointer to class for walking through bw_rle image.

· Bitmap bitmap

The bitmap to be used when painting.

• Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

Additional Inherited Members

7.149.1 Detailed Description

PainterBWBitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.149.2 Constructor & Destructor Documentation

7.149.2.1 PainterBWBitmap()

Constructor.

Parameters

bmp	The bitmap.

7.149.3 Member Function Documentation

7.149.3.1 render()

```
virtual void render (
        uint8_t * ptr,
        int x,
        int xAdjust,
        int y,
        unsigned count,
        const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	X	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterBW.

7.149.3.2 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterBW.

7.149.3.3 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	color	Color of the pixel, 0 or 1.
-----	-------	-----------------------------

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterBW.

7.149.3.4 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp The bitmap.	
-----------------	--

7.150 PainterGRAY2 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/PainterGRAY2.hpp>
```

Public Member Functions

• PainterGRAY2 (colortype color=0, uint8_t alpha=255)

Constructor.

void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

• colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderNext (uint8_t &gray, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterGray

The grey color.

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.150.1 Detailed Description

The PainterGRAY2 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.150.2 Constructor & Destructor Documentation

7.150.2.1 PainterGRAY2()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.150.3 Member Function Documentation

```
7.150.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

```
7.150.3.2 getColor()
```

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.150.3.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	X	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterGRAY2.

7.150.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

ou	t	gray	The gray (0-15).
ou	t	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterGRAY2.

7.150.3.5 setAlpha()

```
void setAlpha (
```

```
uint8_t alpha )
```

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.
-------	-------------------------

7.150.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.151 PainterGRAY2Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterGRAY2Bitmap.hpp>
```

Public Member Functions

- PainterGRAY2Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 - Constructor.
- void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &gray, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapGRAY2Pointer

Pointer to the bitmap (GRAY2)

const uint8 t * bitmapAlphaPointer

Pointer to the bitmap alpha data for GRAY2.

Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.151.1 Detailed Description

PainterGRAY2Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.151.2 Constructor & Destructor Documentation

7.151.2.1 PainterGRAY2Bitmap()

```
PainterGRAY2Bitmap (
            const Bitmap & bmp = Bitmap(BITMAP_INVALID),
            uint8_t alpha = 255 )
```

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.151.3 Member Function Documentation

```
7.151.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.151.3.2 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Χ	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterGRAY2.

7.151.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterGRAY2.

7.151.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	gray	The gray (0-15).
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterGRAY2.

7.151.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

7.151.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

٠	arameters				
	bmp	The bitmap.			

7.152 PainterGRAY4 Class Reference

A Painter that will paint using a color and an alpha value.

```
#include <touchgfx/widgets/canvas/PainterGRAY4.hpp>
```

Public Member Functions

• PainterGRAY4 (colortype color=0, uint8_t alpha=255)

Constructor.

• void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

• colortype getColor () const

Gets the current color.

• void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &gray, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterGray

The grey color.

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.152.1 Detailed Description

The PainterGRAY4 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.152.2 Constructor & Destructor Documentation

7.152.2.1 PainterGRAY4()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.152.3 Member Function Documentation

```
7.152.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.152.3.2 getColor()

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.152.3.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	Х	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterGRAY4.

7.152.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	gray	The gray (0-15).
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterGRAY4.

7.152.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.
-------	-------------------------

7.152.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.153 PainterGRAY4Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterGRAY4Bitmap.hpp>

Public Member Functions

• PainterGRAY4Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

· void setAlpha (uint8 t alpha)

Sets an alpha value for the bitmap.

• uint8 t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &gray, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapGRAY4Pointer

Pointer to the bitmap (GRAY4)

const uint8_t * bitmapAlphaPointer

Pointer to the bitmap alpha data for GRAY4.

· Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.153.1 Detailed Description

PainterGRAY4Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.153.2 Constructor & Destructor Documentation

7.153.2.1 PainterGRAY4Bitmap()

```
PainterGRAY4Bitmap (
            const Bitmap & bmp = Bitmap(BITMAP_INVALID),
            uint8_t alpha = 255 )
```

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.153.3 Member Function Documentation

```
7.153.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.153.3.2 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterGRAY4.

7.153.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterGRAY4.

7.153.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	gray	The gray (0-15).
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterGRAY4.

7.153.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

```
alpha The alpha value to use if there is no alpha channel in the bitmap.
```

7.153.3.6 setBitmap()

```
void setBitmap ( {\tt const\ Bitmap\ \&\ \it bmp\ )}
```

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp	The bitmap.
-----	-------------

7.154 PainterRGB565 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/PainterRGB565.hpp>

Public Member Functions

• PainterRGB565 (colortype color=0, uint8_t alpha=255)

Constructor.

void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

• colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

uint16_t painterColor

The color.

uint16_t painterRed

The red part of the color.

• uint16_t painterGreen

The green part of the color.

uint16_t painterBlue

The blue part of the color.

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.154.1 Detailed Description

The PainterRGB565 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.154.2 Constructor & Destructor Documentation

7.154.2.1 PainterRGB565()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.154.3 Member Function Documentation

```
7.154.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.154.3.2 getColor()

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.154.3.3 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	X	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGB565.

7.154.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGB565.

7.154.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

7.154.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.155 PainterRGB565Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterRGB565Bitmap.hpp>
```

Public Member Functions

- PainterRGB565Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 Constructor.
- void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8 t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

• const uint32_t * bitmapARGB8888Pointer

Pointer to the bitmap (ARGB8888)

const uint16_t * bitmapRGB565Pointer

Pointer to the bitmap (RGB565)

const uint8_t * bitmapAlphaPointer

Pointer to the bitmap alpha data for RGB565.

· Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.155.1 Detailed Description

PainterRGB565Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.155.2 Constructor & Destructor Documentation

7.155.2.1 PainterRGB565Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.155.3 Member Function Documentation

7.155.3.1 getAlpha()

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.155.3.2 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGB565.

7.155.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterRGB565.

7.155.3.4 renderNext()

```
uint8_t & green,
uint8_t & blue,
uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGB565.

7.155.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.
-------	--

7.155.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp The	e bitmap.
---------	-----------

7.156 PainterRGB565L8Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterRGB565L8Bitmap.hpp>
```

Public Member Functions

• PainterRGB565L8Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

 $\bullet \ \ \text{virtual void } \\ \text{render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)} \\$

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

• const uint8_t * bitmapPointer

Pointer to the bitmap (L8)

const uint8_t * bitmapExtraPointer

Pointer to the bitmap alpha data for RGB565 / CLUT for L8.

· Bitmap bitmap

The bitmap to be used when painting.

• Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.156.1 Detailed Description

PainterRGB565L8Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.156.2 Constructor & Destructor Documentation

7.156.2.1 PainterRGB565L8Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.156.3 Member Function Documentation

```
7.156.3.1 getAlpha()
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

The current alpha value.

See also

Returns

setAlpha

7.156.3.2 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGB565.

7.156.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterRGB565.

7.156.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGB565.

7.156.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

```
alpha The alpha value to use if there is no alpha channel in the bitmap.
```

7.156.3.6 setBitmap()

```
void setBitmap ( {\tt const\ Bitmap\ \&\ \it bmp\ )}
```

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

7.157 PainterRGB888 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/PainterRGB888.hpp>

Public Member Functions

• PainterRGB888 (colortype color=0, uint8_t alpha=255)

Constructor.

• void setColor (colortype color, uint8_t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

• uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

uint8_t painterRed

The red part of the color.

• uint8_t painterGreen

The green part of the color.

uint8_t painterBlue

The blue part of the color.

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.157.1 Detailed Description

The PainterRGB888 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.157.2 Constructor & Destructor Documentation

7.157.2.1 PainterRGB888()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.157.3 Member Function Documentation

```
7.157.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.157.3.2 getColor()

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.157.3.3 render()

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGB888.

7.157.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

 $Implements\ Abstract Painter RGB888.$

7.157.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.
-------	-------------------------

7.157.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.158 PainterRGB888Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterRGB888Bitmap.hpp>

Public Member Functions

- PainterRGB888Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 - Constructor
- · void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint32_t * bitmapARGB8888Pointer

Pointer to the bitmap (ARGB8888)

const uint8_t * bitmapRGB888Pointer

Pointer to the bitmap (RGB888)

Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

uint8_t painterAlpha

The alpha to use combined with alpha data from the bitmap.

Additional Inherited Members

7.158.1 Detailed Description

PainterRGB888Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.158.2 Constructor & Destructor Documentation

7.158.2.1 PainterRGB888Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.158.3 Member Function Documentation

```
7.158.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.158.3.2 render()

```
virtual void render (
          uint8_t * ptr,
          int x,
          int xAdjust,
          int y,
          unsigned count,
          const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
	X	The x coordinate.	
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterRGB888.

7.158.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterRGB888.

7.158.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGB888.

7.158.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.
-------	--

7.158.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp	The bitmap.

7.159 PainterRGB888L8Bitmap Class Reference

A Painter that will paint using a bitmap.

```
#include <touchgfx/widgets/canvas/PainterRGB888L8Bitmap.hpp>
```

Public Member Functions

- PainterRGB888L8Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)
 - Constructor.
- void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

• void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8 t &red, uint8 t &green, uint8 t &blue, uint8 t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8 t * bitmapPointer

Pointer to the bitmap (L8)

const uint8_t * bitmapExtraPointer

Pointer to the CLUT (L8)

Bitmap bitmap

The bitmap to be used when painting.

Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8 t painterAlpha

The alpha to use combined with alpha data from the bitmap.

Additional Inherited Members

7.159.1 Detailed Description

PainterRGB888L8Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.159.2 Constructor & Destructor Documentation

7.159.2.1 PainterRGB888L8Bitmap()

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.159.3 Member Function Documentation

```
7.159.3.1 getAlpha()
uint8_t getAlpha ( ) const
Gets the current alpha value.
```

Returns

The current alpha value.

See also

setAlpha

7.159.3.2 render()

```
virtual void render (
        uint8_t * ptr,
        int x,
        int xAdjust,
        int y,
        unsigned count,
        const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Χ	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGB888.

```
7.159.3.3 renderInit()
```

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterRGB888.

7.159.3.4 renderNext()

```
virtual bool renderNext (
          uint8_t & red,
          uint8_t & green,
          uint8_t & blue,
          uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGB888.

7.159.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha	The alpha value to use if there is no alpha channel in the bitmap.
-------	--

7.159.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

_		
ſ	bmp	The bitmap.

7.160 PainterRGBA2222 Class Reference

A Painter that will paint using a color and an alpha value.

#include <touchgfx/widgets/canvas/PainterRGBA2222.hpp>

Public Member Functions

• PainterRGBA2222 (colortype color=0, uint8 t alpha=255)

Constructor.

void setColor (colortype color, uint8 t alpha=255)

Sets color and alpha to use when drawing the CanvasWidget.

• colortype getColor () const

Gets the current color.

void setAlpha (uint8_t alpha)

Sets an alpha value for the painter.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

· uint8_t painterColor

The color.

uint8_t painterRed

The red part of the color, scaled up to [0..255].

uint8_t painterGreen

The green part of the color, scaled up to [0..255].

uint8_t painterBlue

The blue part of the color, scaled up to [0..255].

· uint8_t painterAlpha

The alpha value.

Additional Inherited Members

7.160.1 Detailed Description

The PainterRGBA2222 class allows a shape to be filled with a given color and alpha value. This allows transparent, anti-aliased elements to be drawn.

See also

AbstractPainter

7.160.2 Constructor & Destructor Documentation

7.160.2.1 PainterRGBA2222()

Constructor.

Parameters

color	the color.
alpha	the alpha.

7.160.3 Member Function Documentation

```
7.160.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

```
7.160.3.2 getColor()
```

```
colortype getColor ( ) const
```

Gets the current color.

Returns

The color.

7.160.3.3 render()

```
int y,
unsigned count,
const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.
	Х	The x coordinate.
	xAdjust	The minor adjustment of x (used when a pixel is smaller than a byte to specify that the pointer should have been advanced "xAdjust" pixels futher).
	У	The y coordinate.
	count	Number of pixels to fill.
	covers	The coverage in of each pixel.

Reimplemented from AbstractPainterRGBA2222.

7.160.3.4 renderNext()

```
virtual bool renderNext (
            uint8_t & red,
            uint8_t & green,
            uint8_t & blue,
            uint8_t & alpha ) [protected], [virtual]
```

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGBA2222.

7.160.3.5 setAlpha()

Sets an alpha value for the painter.

Parameters

alpha	The alpha value to use.			

7.160.3.6 setColor()

Sets color and alpha to use when drawing the CanvasWidget.

Parameters

color	The color.
alpha	The alpha.

7.161 PainterRGBA2222Bitmap Class Reference

A Painter that will paint using a bitmap.

#include <touchgfx/widgets/canvas/PainterRGBA2222Bitmap.hpp>

Public Member Functions

• PainterRGBA2222Bitmap (const Bitmap &bmp=Bitmap(BITMAP_INVALID), uint8_t alpha=255)

Constructor.

void setBitmap (const Bitmap &bmp)

Sets a bitmap to be used when drawing the CanvasWidget.

void setAlpha (uint8_t alpha)

Sets an alpha value for the bitmap.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void render (uint8_t *ptr, int x, int xAdjust, int y, unsigned count, const uint8_t *covers)

Paint a designated part of the RenderingBuffer.

Protected Member Functions

virtual bool renderInit ()

Initialize rendering of a single scan line of pixels for the render.

• virtual bool renderNext (uint8_t &red, uint8_t &green, uint8_t &blue, uint8_t &alpha)

Get the color of the next pixel in the scan line.

Protected Attributes

const uint8_t * bitmapRGBA2222Pointer

Pointer to the bitmap (RGBA2222)

· Bitmap bitmap

The bitmap to be used when painting.

• Rect bitmapRectToFrameBuffer

Bitmap rectangle translated to frame buffer coordinates.

· uint8_t painterAlpha

The alpha to use if no alpha data is present in the given bitmap.

Additional Inherited Members

7.161.1 Detailed Description

PainterRGBA2222Bitmap will take the color for a given point in the shape from a bitmap. Please be aware, the the bitmap is used by the CWR (not Shape), so any rotation you might specify for a Canvas Widget (e.g. Shape) is not applied to the bitmap as CWR is not aware of this rotation.

See also

AbstractPainter

7.161.2 Constructor & Destructor Documentation

7.161.2.1 PainterRGBA2222Bitmap()

```
PainterRGBA2222Bitmap (
            const Bitmap & bmp = Bitmap(BITMAP_INVALID),
            uint8_t alpha = 255 )
```

Constructor.

Parameters

bmp	The bitmap.
alpha	the alpha.

7.161.3 Member Function Documentation

```
7.161.3.1 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.161.3.2 render()

```
int xAdjust,
int y,
unsigned count,
const uint8_t * covers ) [virtual]
```

Paint a designated part of the RenderingBuffer with respect to the amount of coverage of each pixel given by the parameter covers.

Parameters

in	ptr	Pointer to the row in the RenderingBuffer.	
x The x coordinate.		The x coordinate.	
	xAdjust	st The minor adjustment of x (used when a pixel is smaller than a byte to specify that the point should have been advanced "xAdjust" pixels futher).	
	У	The y coordinate.	
	count	Number of pixels to fill.	
	covers	The coverage in of each pixel.	

Reimplemented from AbstractPainterRGBA2222.

7.161.3.3 renderInit()

```
virtual bool renderInit ( ) [protected], [virtual]
```

Initialize rendering of a single scan line of pixels for the render.

Returns

true if it succeeds, false if it fails.

Reimplemented from AbstractPainterRGBA2222.

7.161.3.4 renderNext()

Get the color of the next pixel in the scan line.

Parameters

out	red	The red.
out	green	The green.
out	blue	The blue.
out	alpha	The alpha.

Returns

true if the pixel should be painted, false otherwise.

Implements AbstractPainterRGBA2222.

7.161.3.5 setAlpha()

Sets an alpha value for the bitmap. If the image contains an alpha channel, this alpha value is combined with the alpha in the bitmap to produce the final alpha value.

Parameters

alpha The alpha value to use if there is no alpha channel in t	the bitmap.
--	-------------

7.161.3.6 setBitmap()

Sets a bitmap to be used when drawing the CanvasWidget.

Parameters

bmp The bitmap.

7.162 Pair < T1, T2 > Struct Template Reference

A simple struct for holding pairs of data.

```
#include <touchgfx/hal/Types.hpp>
```

Public Member Functions

• Pair ()

Default constructor.

• Pair (const T1 &x, const T2 &y)

Constructor initializing the elements it holds using their copy constructor.

 • template < class U , class V > Pair (const Pair < U, V > &p)

Copy constructor.

Public Attributes

T1 first

The first element.

• T2 second

The second element.

7.162.1 Detailed Description

```
template < class T1, class T2> struct touchgfx::Pair < T1, T2>
```

A simple struct for holding pairs of data.

Template Parameters

T1	The type of the first element.	
T2	The type of the second element.	

7.162.2 Constructor & Destructor Documentation

```
7.162.2.1 Pair() [1/3]
Pair ( ) [inline]
```

Constructor initializing the elements it holds using their default constructors.

Constructor initializing the elements it holds using their copy constructor.

Parameters

X	Reference to the first element.
У	Reference to the second element.

Copy constructor.

Template Parameters

U	Generic type parameter.
V	Generic type parameter.

Parameters

n	The pair to copy from.
~	i i i i i i i i i i i i i i i i i i i

7.163 Partition < ListOfTypes, NUMBER_OF_ELEMENTS > Class Template Reference

This type provides a concrete Partition of memory-slots capable of holding any of the specified list of types.

```
#include <common/Partition.hpp>
```

Public Types

- enum { INTS_PR_ELEMENT = (sizeof(typename meta::select_type_maxsize < SupportedTypesList >::type) + sizeof(int) 1) / sizeof(int), SIZE_OF_ELEMENT = INTS_PR_ELEMENT * sizeof(int) }
- typedef ListOfTypes SupportedTypesList

Provides a generic public type containing the list of supported types.

Public Member Functions

• Partition ()

Default constructor.

virtual ∼Partition ()

Destructor.

virtual uint16_t capacity () const

Specialization of AbstractPartition::capacity().

• virtual uint32_t element_size ()

Specialization of AbstractPartition::element_size().

Protected Member Functions

• virtual void * element (uint16_t index)

Specialization of AbstractPartition::element()

virtual const void * element (uint16_t index) const

Specialization of AbstractPartition::element() const.

7.163.1 Detailed Description

```
template<typename ListOfTypes, uint16_t NUMBER_OF_ELEMENTS> class touchgfx::Partition< ListOfTypes, NUMBER_OF_ELEMENTS>
```

The Partition is not aware of the types stored in the Partition memory, hence it provides no mechanism for deleting C++ objects when the Partition is clear()'ed.

This class implements AbstractPartition.

Template Parameters

ListOfTypes	Type of the list of types.
NUMBER_OF_ELEMENTS	Type of the number of elements.

See also

AbstractPartition

7.163.2 Member Typedef Documentation

```
7.163.2.1 SupportedTypesList
```

```
ListOfTypes SupportedTypesList
```

Provides a generic public type containing the list of supported types.

7.163.3 Member Enumeration Documentation

```
7.163.3.1 anonymous enum
```

```
anonymous enum
```

Compile-time generated constants specifying the "element" or "slot" size used by this partition

7.163.4 Constructor & Destructor Documentation

```
7.163.4.1 Partition()
```

```
Partition ( ) [inline]
```

Constructs an empty Partition.

7.163.5 Member Function Documentation

```
7.163.5.1 capacity()
```

```
uint16_t capacity ( ) const [inline], [virtual]
```

Specialization of AbstractPartition::capacity().

Returns

An uint16_t.

See also

touchgfx::AbstractPartition::capacity()

Implements AbstractPartition.

```
7.163.5.2 element() [1/2]
void * element (
              uint16_t index ) [inline], [protected], [virtual]
Parameters
 index
         Zero-based index of the.
Returns
     null if it fails, else a void*.
See also
     touchgfx::AbstractPartition::element()
Implements AbstractPartition.
7.163.5.3 element() [2/2]
const void * element (
               uint16_t index ) const [inline], [protected], [virtual]
Parameters
 index Zero-based index of the.
Returns
     null if it fails, else a void*.
See also
     touchgfx::AbstractPartition::element()
Implements AbstractPartition.
7.163.5.4 element_size()
uint32_t element_size ( ) [inline], [virtual]
Specialization of AbstractPartition::element_size().
Returns
     An uint32_t.
See also
     touchgfx::AbstractPartition::element_size()
```

Implements AbstractPartition.

7.164 PixelDataWidget Class Reference

A widget for displaying a buffer of pixel data.

#include <touchgfx/widgets/PixelDataWidget.hpp>

Public Member Functions

· PixelDataWidget ()

Default constructor.

virtual void draw (const touchgfx::Rect &invalidatedArea) const

Draw the part of the RAM buffer that is inside the invalidated area.

virtual touchgfx::Rect getSolidRect () const

Report this widget as being completely solid.

void setPixelData (uint8_t *const data)

Set the pixel data to display.

· void setBitmapFormat (Bitmap::BitmapFormat format)

Set the format of the pixel data.

void setAlpha (uint8_t a)

Sets the alpha channel for the image.

• uint8_t getAlpha () const

Gets the current alpha value.

Protected Attributes

uint8 t * buffer

The buffer where the pixels are copied from.

• Bitmap::BitmapFormat format

The pixel format for the data.

· uint8 t alpha

The Alpha for this widget.

Additional Inherited Members

7.164.1 Detailed Description

The buffer must be of size as widget. If the LCD is 16 bit the buffer must hold 2 bytes for each pixel. If the LCD is 24 bit the buffer must hold 3 bytes for each pixel.

See also

touchgfx::Widget

7.164.2 Constructor & Destructor Documentation

7.164.2.1 PixelDataWidget()

PixelDataWidget ()

Default constructor.

7.164.3 Member Function Documentation

Draw the part of the RAM buffer that is inside the invalidated area.

Parameters

invalidatedArea	The region of this drawable that needs to be redrawn.
-----------------	---

See also

touchgfx::Drawable

Implements Drawable.

```
7.164.3.2 getAlpha()
```

```
uint8_t getAlpha ( ) const
```

Gets the current alpha value.

Returns

The current alpha value.

See also

setAlpha

7.164.3.3 getSolidRect()

```
touchgfx::Rect getSolidRect ( ) const [virtual]
```

Report this widget as being completely solid.

Returns

The solid rect.

Implements Drawable.

7.164.3.4 setAlpha()

Sets the alpha channel for the image.

Parameters

```
a The alpha value. 255 = completely solid.
```

7.164.3.5 setBitmapFormat()

Set the format of the pixel data.

Parameters

escribes the format to use.	format	
-----------------------------	--------	--

7.164.3.6 setPixelData()

Set the pixel data to display.

Parameters

in,out	data	If non-null, the data.
--------	------	------------------------

7.165 Point Struct Reference

A simple struct containing coordinates.

```
#include <touchgfx/hal/Types.hpp>
```

Public Member Functions

• unsigned dist_sqr (struct Point &o)

The squared distance from this Point to another Point.

Public Attributes

```
    int32 t x
```

The x coordinate.

int32_t y

The y coordinate.

7.165.1 Member Function Documentation

7.165.1.1 dist_sqr()

The squared distance from this Point to another Point.

Parameters

in	0	The point to get the squared distance to.
----	---	---

Returns

The squared distance.

7.166 Point3D Struct Reference

A 3D point.

```
#include <touchgfx/hal/Types.hpp>
```

Public Attributes

• fixed28_4 X

The X coordinate.

fixed28_4 Y

The Y coordinate.

float Z

The Z coordinate.

float U

The U coordinate.

float V

The V coordinate.

7.167 Point4 Class Reference

This class represents a homogeneous 3D point.

```
#include <touchgfx/Math3D.hpp>
```

Public Member Functions

• Point4 ()

Default constructor.

• Point4 (float x, float y, float z)

Constructor.

Additional Inherited Members

7.167.1 Detailed Description

This class represents a homogeneous 3D point.

See also

quadruple

7.167.2 Constructor & Destructor Documentation

Constructor.

Parameters

X	The x value.
У	The y value.
Z	The z value.

7.168 PreRenderable < T > Class Template Reference

This mixin can be used on any Drawable.

```
#include <touchgfx/mixins/PreRenderable.hpp>
```

Public Member Functions

• PreRenderable ()

Default constructor.

• void draw (const Rect &invalidatedArea) const

Overrides the draw function.

• virtual void setupDrawChain (const Rect &invalidatedArea, Drawable **nextPreviousElement)

Add to draw chain.

• bool isPreRendered () const

Whether or not the snapshot of the widget been taken.

• void preRender ()

Takes a snapshot of the current visual appearance of this widget.

7.168.1 Detailed Description

```
template < class T> class touchgfx::PreRenderable < T >
```

This mixin can be used on any Drawable. It provides a preRender function, which will cache the current visual appearance of the Drawable to be cache in a memory region. Subsequent calls to draw() on this Drawable will result in a simple memory of the cached memory instead of the normal draw operation. This mixin can therefore be used on Drawables whose visual appearance is static and the normal draw operation takes a long time to compute.

Note

The actual uses of this mixin are rare, and the class is mainly provided for example purposes.

Template Parameters

```
T The type of Drawable to add this functionality to.
```

7.168.2 Constructor & Destructor Documentation

7.168.2.1 PreRenderable()

```
PreRenderable ( ) [inline]
```

Default constructor. Initializes the PreRenderable.

7.168.3 Member Function Documentation

7.168.3.1 draw()

Overrides the draw function. If preRender has been called, perform a memcpy of the cached version. If not, just call the base class version of draw.

Parameters

```
invalidatedArea The subregion of this Drawable which needs to be redrawn.
```

7.168.3.2 isPreRendered()

```
bool isPreRendered ( ) const [inline]
```

Returns

Is the widget rendered.

7.168.3.3 preRender()

```
void preRender ( ) [inline]
```

Takes a snapshot of the current visual appearance of this widget. All subsequent calls to draw on this mixin will result in the snapshot being draw.

7.168.3.4 setupDrawChain()

Note

For TouchGFX internal use only.

Parameters

	invalidatedArea	Include drawables that intersect with this area only.
in,out	nextPreviousElement	Modifiable element in linked list.

7.169 Presenter Class Reference

The Presenter base class that all application-specific presenters should derive from.

```
#include <mvp/Presenter.hpp>
```

Public Member Functions

• virtual void activate ()

Place initialization code for the Presenter here.

• virtual void deactivate ()

Place cleanup code for the Presenter here.

• virtual \sim Presenter ()

Destructor.

Protected Member Functions

• Presenter ()

Default constructor.

7.169.1 Detailed Description

The Presenter base class that all application-specific presenters should derive from. Only contains activate and deactivate virtual functions which are called automatically during screen transition.

7.169.2 Constructor & Destructor Documentation

7.169.2.1 ∼Presenter()

```
~Presenter ( ) [inline], [virtual]
```

Destructor.

7.169.2.2 Presenter()

```
Presenter ( ) [inline], [protected]
```

Default constructor.

7.169.3 Member Function Documentation

7.169.3.1 activate()

```
void activate ( ) [inline], [virtual]
```

The activate function is called automatically when a screen transition causes this Presenter to become active. Place initialization code for the Presenter here.

7.169.3.2 deactivate()

```
void deactivate ( ) [inline], [virtual]
```

The deactivate function is called automatically when a screen transition causes this Presenter to become inactive. Place cleanup code for the Presenter here.

7.170 CWRUtil::Q10 Class Reference

Defines a number with 10 bits reserved for fraction.

```
#include <touchgfx/widgets/canvas/CWRUtil.hpp>
```

Public Member Functions

• Q10 ()

Default constructor.

• Q10 (int i)

Constructor from integer.

operator int () const

Gets the Q10 as an integer without conversion.

• Q10 operator- () const

Negation operator.

• Q10 operator+ (const Q10 &q10) const

Addition operator.

Q15 operator* (const Q5 &q5) const

Multiplication operator.

• Q5 operator/ (const Q5 &q5) const

Division operator.

```
    template<typename T >
        T to () const

    Converts the Q10 value to an int or a float.
```

7.170.1 Detailed Description

Defines a number with 10 bits reserved for the fractional part of the decimal number. Q10 implements some simple arithmetic operations, most yielding a Q10 number and some yielding a Q5 number or a Q15 number as a result.

```
Q5*Q5=Q10, Q10/Q5=Q5, ...
```

See also

Q5 Q15

```
http://en.wikipedia.org/wiki/Q_%28number_format%29
http://en.wikipedia.org/wiki/Fixed-point_arithmetic
```

7.170.2 Constructor & Destructor Documentation

Constructor from integer. No conversion is done - the integer is assumed to already be in Q10 format.

Parameters

```
i int pre-formattet in Q10 format.
```

7.170.3 Member Function Documentation

```
7.170.3.1 operator int()
operator int ( ) const [inline]
```

Gets the Q10 as an integer without conversion.

Returns

The unconverted Q10 value.

7.170.3.2 operator*()

```
Q15 operator* ( {\tt const~Q5~\&~q5~)~const~[inline]}
```

Multiplication operator. The result is a Q15, not a Q10, for increased precision.

Parameters

```
q5 The Q5 to multiply this with.
```

Returns

The result of the operation.

7.170.3.3 operator+()

```
Q10 operator+ ( {\tt const~Q10~\&~q10~)~const~[inline]}
```

Addition operator.

Parameters

q10	The Q10 to add to this.
-----	-------------------------

Returns

The result of the operation.

7.170.3.4 operator-()

```
Q10 operator- ( ) const [inline]
```

Negation operator.

Returns

The negative value of this.

7.170.3.5 operator/()

```
Q5 operator/ ( {\rm const} \ {\rm Q5} \ \& \ q5 \ ) \ {\rm const} \ [{\rm inline}]
```

Division operator.

Parameters

```
q5 The Q5 to divide this by.
```

Returns

The result of the operation.

7.170.3.6 to()

```
template< typename T > T to ( ) const [inline]
```

Convert the Q10 value to an integer by removing the 10 bits used for the fraction, or to a floating point value by dividing by 32 * 32, depending on the type specified as T.

Template Parameters

T Either int or float.

Returns

Q10 value as a type T.

7.171 CWRUtil::Q15 Class Reference

Defines a number with 15 bits reserved for fraction.

```
#include <touchgfx/widgets/canvas/CWRUtil.hpp>
```

Public Member Functions

• Q15 (int i)

Constructor from integer. No conversion is done - the integer is assumed to already be in Q15 format.

· operator int () const

Gets the Q15 as an integer without conversion.

• Q15 operator- () const

Negation operator.

• Q15 operator+ (const Q15 &q15) const

Addition operator.

• Q10 operator/ (const Q5 &q5) const

Q5 / Q5 which requires the result of a Q15 / Q5 to be calculated.

7.171.1 Detailed Description

Defines a number with 15 bits reserved for the fractional part of the decimal number. Q15 is only used for sine/cosine and for intermediate calculations when multiplying.

```
Q5*Q5=Q10, Q10/Q5=Q5, ...
```

See also

Q5 Q10

```
http://en.wikipedia.org/wiki/Q_%28number_format%29
http://en.wikipedia.org/wiki/Fixed-point_arithmetic
```

7.171.2 Constructor & Destructor Documentation

```
7.171.2.1 Q15()
Q15 (
         int i ) [inline], [explicit]
```

Constructor from integer. No conversion is done - the integer is assumed to already be in Q15 format.

Parameters

```
i int pre-formattet in Q15 format.
```

7.171.3 Member Function Documentation

```
7.171.3.1 operator int()
operator int ( ) const [inline]
```

Gets the Q15 as an integer without conversion.

Returns

The unconverted Q15 value.

7.171.3.2 operator+()

```
Q15 operator+ (  {\rm const} \ {\rm Q15} \ \& \ q15 \ ) \ {\rm const} \ \ [{\rm inline}]
```

Addition operator.

Parameters

```
q15 The Q10 to add to this.
```

Returns

The result of the operation.

7.171.3.3 operator-()

```
Q15 operator- ( ) const [inline]
```

Negation operator.

Returns

The negative value of this.

7.171.3.4 operator/()

```
Q10 operator/ ( {\tt const~Q5~\&~q5~)~const~[inline]}
```

Q5 / Q5 which requires the result of a Q15 / Q5 to be calculated.

Parameters

q5 The Q5 to divide this by.

Returns

The result of the operation.

7.172 CWRUtil::Q5 Class Reference

Defines a number with 5 bits reserved for fraction.

```
#include <touchgfx/widgets/canvas/CWRUtil.hpp>
```

Public Member Functions

• Q5 ()

Default constructor.

• Q5 (int i)

Constructor from integer.

• Q5 (const Q10 q10)

Constructor from Q10.

· operator int () const

Gets the Q5 as an integer without conversion.

• Q5 operator- () const

Negation operator.

• Q5 operator+ (const Q5 &q5) const

Addition operator.

Q5 operator- (const Q5 &q5) const

Subtraction operator.

• Q10 operator* (const Q5 &q5) const

Multiplication operator.

Q5 operator* (const Q15 &q15) const

Multiplication operator.

• Q5 operator* (const int i) const

Multiplication operator.

• Q5 operator/ (const int i) const

Division operator.

• Q5 operator/ (const Q5 q5) const

Division operator.

```
    template < typename T >
        T to () const

    Converts the Q5 value to an int or a float.
```

7.172.1 Detailed Description

Defines a number with 5 bits reserved for the fractional part of the decimal number. Q5 implements some simple arithmetic operations, most yielding a Q5 number and some yielding a Q10 number as a result. Other operations also work with Q15 numbers.

See also

```
Q10
Q15
http://en.wikipedia.org/wiki/Q_%28number_format%29
http://en.wikipedia.org/wiki/Fixed-point_arithmetic
```

7.172.2 Constructor & Destructor Documentation

```
7.172.2.1 Q5() [1/2] Q5 ( \inf i ) \quad [inline], \ [explicit]
```

Constructor from integer. No conversion is done - the integer is assumed to already be in Q5 format.

Parameters

i Integer pre-formattet in Q5 format.

Constructor from Q10. The Q10 is shifted down to convert it to Q5.

Parameters

```
q10 The Q10 value to convert to a Q5 value.
```

See also

Q10

7.172.3 Member Function Documentation

7.172.3.1 operator int()

```
operator int ( ) const [inline]
```

Gets the Q5 as an integer without conversion.

Returns

The unconverted Q5 value.

Multiplication operator. The result is a Q10, not a Q5, for increased precision.

Parameters

```
q5 The Q5 to multiply this with.
```

Returns

The result of the operation.

See also

Q10

Multiplication operator. Often used in relation with sine and cosine calculation which are pre-calculated as Q15. As the result is needed as a Q5, this operator multiplies with the given Q15 and converts the result to a Q5.

Parameters

```
q15 The Q15 to multiply this with.
```

Returns

The result of the operation.

See also

Q15

Multiplication operator.

Parameters

```
i The integer to multiply this with.
```

Returns

The result of the operation.

7.172.3.5 operator+()

Addition operator.

Parameters

```
q5 The Q5 to add to this.
```

Returns

The result of the operation.

```
7.172.3.6 operator-() [1/2]
Q5 operator- ( ) const [inline]
```

Negation operator.

Returns

The negative value of this.

Subtraction operator.

Parameters

```
q5 The Q5 to subtract from this.
```

Returns

The result of the operation.

Division operator.

Parameters

i The integer to divide this by.

Returns

The result of the operation.

```
7.172.3.9 operator/() [2/2]  \begin{tabular}{ll} Q5 & operator/ ( & const & Q5 & q5 \end{tabular} ) & const & [inline] \\ \end{tabular}
```

Division operator. Internally this Q5 is converted to Q10 before the division to increased precision.

Parameters

```
q5 The Q5 to divide this by.
```

Returns

The result of the operation.

See also

Q10

7.172.3.10 to()

```
template< typename T > T to ( ) const [inline]
```

Convert the Q5 value to an integer by removing the 5 bits used for the fraction, or to a floating point value by dividing by 32, depending on the type specified as T.

Template Parameters

T Either int or float.

Returns

Q5 value as a type T.

7.173 Quadruple Class Reference

Base class for homogeneous vectors and points.

#include <touchgfx/Math3D.hpp>

Public Member Functions

• float getElement (int row) const

Gets an element.

· float getX () const

Get x coordinate.

• float getY () const

Get y coordinate.

· float getZ () const

Get z coordinate.

• float getW () const

Gets the w.

void setElement (int row, float value)

Sets an element.

void setX (float value)

Sets an x coordinate.

void setY (float value)

Sets a y coordinate.

void setZ (float value)

Sets a z coordinate.

• void setW (float value)

Sets a w.

Protected Member Functions

• Quadruple ()

Default constructor.

Quadruple (float x, float y, float z, float w)

Constructor.

Protected Attributes

· float elements [4]

The elements[4].

7.173.1 Detailed Description

Base class for homogeneous vectors and points.

7.173.2 Constructor & Destructor Documentation

Constructor.

Parameters

X	The x value.
У	The y value.
Z	The z value.
W	The w value.

7.173.3 Member Function Documentation

```
7.173.3.1 getElement()
```

Gets an element.

Parameters

```
row The row.
```

Returns

The element.

7.173.3.2 getW()

```
float getW ( ) const [inline]
```

Gets the w.

Returns

The w.

```
7.173.3.3 getX()
```

```
float getX ( ) const [inline]
```

Get x coordinate.

Returns

The x coordinate.

```
7.173.3.4 getY()
```

```
float getY ( ) const [inline]
```

Get y coordinate.

Returns

The y coordinate.

7.173.3.5 getZ()

```
float getZ ( ) const [inline]
```

Get z coordinate.

Returns

The z coordinate.

7.173.3.6 setElement()

Sets an element.

Parameters

row	The row.
value	The value.

```
7.173.3.7 setW()
```

```
void setW ( \label{eq:float_value} \mbox{float } \mbox{\it value} \mbox{ ) } \mbox{ [inline]}
```

Sets a w.

Parameters

value The value.

7.173.3.8 setX()

Sets an x coordinate.

Parameters

value The value.

7.173.3.9 setY()

Sets a y coordinate.

Parameters

value The value.

7.173.3.10 setZ()

Sets a z coordinate.

Parameters

value The value.

7.174 RadioButton Class Reference

Radio button with two states.

#include <touchgfx/widgets/RadioButton.hpp>

Public Member Functions

· RadioButton ()

Default constructor.

virtual ∼RadioButton ()

Destructor.

virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

virtual void handleClickEvent (const ClickEvent &event)

Updates the current state of the radio button.

• virtual void setBitmaps (const Bitmap &bmpUnselected, const Bitmap &bmpUnselectedPressed, const Bitmap &bmpSelected, const Bitmap &bmpSelectedPressed)

Sets the bitmaps used by this button.

void setDeselectedAction (GenericCallback< const AbstractButton & > &callback)

Associates an action to be performed when the AbstractButton is deselected.

· virtual Rect getSolidRect () const

Gets solid rectangle.

· void setAlpha (uint8_t alpha)

Sets the alpha channel for the image.

• uint8_t getAlpha () const

Gets the current alpha value.

void setDeselectionEnabled (bool state)

States whether or not it is possible to de-select the RadioButton by clicking it.

• bool getDeselectionEnabled () const

Gets the current deselectionEnabled state.

void setSelected (bool newSelected)

Sets the radio buttons selected state.

• bool getSelected () const

Gets the current selected state.

• Bitmap getCurrentlyDisplayedBitmap () const

Gets currently displayed bitmap.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

· Bitmap bitmapUnselected

The image to display when radio button unselected.

Bitmap bitmapUnselectedPressed

The image to display when radio button unselected and pressed.

· Bitmap bitmapSelected

The image to display when radio button selected.

Bitmap bitmapSelectedPressed

The image to display when radio button selected and pressed.

• uint8_t alpha

The current alpha value. 255 denotes solid, 0 denotes completely transparent.

· bool selected

The current selected state.

· bool deselectionEnabled

Is de-selecting a selected radio button by clicking it enabled.

GenericCallback< const AbstractButton &> * deselectedAction

The callback to be executed when this AbstractButton is unselected.

Additional Inherited Members

7.174.1 Detailed Description

A radio button consists of four images, one for its not selected and one for selected. Each of these have an image for a pressed state. RadioButtons can be added to a RadioButtonGroup which handles the de-selection of radio buttons when a new selection is made.

See also

AbstractButton

7.174.2 Constructor & Destructor Documentation

```
7.174.2.1 RadioButton()
RadioButton ( ) [inline]
Default constructor.
7.174.2.2 ~RadioButton()
~RadioButton ( ) [inline], [virtual]
```

7.174.3 Member Function Documentation

Draws the given invalidated area.

Parameters

Destructor.

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.

See also

Drawable::draw()

Gets the current alpha value.

Implements Drawable.

```
7.174.3.2 getAlpha()
uint8_t getAlpha ( ) const [inline]
```

Returns

The current alpha value.

7.174.3.3 getCurrentlyDisplayedBitmap()

```
Bitmap getCurrentlyDisplayedBitmap ( ) const [inline]
```

Function to obtain the currently displayed bitmap, which depends on the radio button's pressed and selected state.

Returns

The bitmap currently displayed.

7.174.3.4 getDeselectionEnabled()

```
bool getDeselectionEnabled ( ) const [inline]
```

Gets the current deselectionEnabled state.

Returns

The current deselectionEnabled state.

7.174.3.5 getSelected()

```
bool getSelected ( ) const [inline]
```

Gets the current selected state.

Returns

The current selected state.

7.174.3.6 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets solid rectangle.

Returns

largest possible solid rect. Delegated to the largest solid rect of the radio button bitmap(s).

Implements Drawable.

7.174.3.7 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE RADIOBUTTON.

Reimplemented from AbstractButton.

7.174.3.8 handleClickEvent()

Updates the current state of the radio button - pressed or released, selected or not selected - and invalidates it.

If a transition from the not selected to selected was made, the associated action is executed and then the Widget is invalidated.

Parameters

event	Information about the click.
-------	------------------------------

See also

Drawable::handleClickEvent()

Reimplemented from AbstractButton.

7.174.3.9 setAlpha()

Sets the alpha channel for the image.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.174.3.10 setBitmaps()

Sets the bitmaps used by this button. If no special pressed states are needed just specify the same bitmap for both pressed and non-pressed bitmaps.

Parameters

bmpUnselected	Bitmap to use when button is unselected.
bmpUnselectedPressed	Bitmap to use when button is unselected and pressed.
bmpSelected	Bitmap to use when button is selected.
bmpSelectedPressed	Bitmap to use when button is selected and pressed.

7.174.3.11 setDeselectedAction()

Associates an action to be performed when the AbstractButton is deselected.

Parameters

See also

GenericCallback

7.174.3.12 setDeselectionEnabled()

States whether or not it is possible to de-select the RadioButton by clicking it.

Parameters

state true if it should be possible to de-select by click.

7.174.3.13 setSelected()

Sets the radio buttons selected state.

Parameters

newSelected the new selected state.

7.175 RadioButtonGroup < CAPACITY > Class Template Reference

Class for handling a collection of RadioButtons.

#include <touchgfx/widgets/RadioButtonGroup.hpp>

Public Member Functions

RadioButtonGroup ()

Default constructor.

virtual ∼RadioButtonGroup ()

Destructor.

virtual void add (RadioButton &radioButton)

Add the RadioButton to the RadioButtonGroup.

virtual RadioButton * getRadioButton (uint16_t index) const

Gets the RadioButton at the specified index.

virtual int32_t getSelectedRadioButtonIndex () const

Gets the index of the selected RadioButton.

• virtual RadioButton * getSelectedRadioButton () const

Gets the selected RadioButton.

virtual void setSelected (RadioButton &radioButton)

Sets the specified RadioButton to be selected.

• virtual void setDeselectionEnabled (bool deselectionEnabled)

Sets whether or not it is possible to deselect RadioButtons by clicking them when they are selected.

virtual bool getDeselectionEnabled () const

Gets the current deselectionEnabled state.

void setRadioButtonSelectedHandler (GenericCallback< const AbstractButton & > &callback)

Associate an action with a radio button.

void setRadioButtonDeselectedHandler (GenericCallback< const AbstractButton & > &callback)

Associate an action with a radio button.

Protected Member Functions

virtual void radioButtonClickedHandler (const AbstractButton &radioButton)

Handles the event that a RadioButton has been selected.

virtual void radioButtonDeselectedHandler (const AbstractButton &radioButton)

Handles the event that a RadioButton has been deselected.

Protected Attributes

• RadioButton * radioButtons [CAPACITY]

The list of added RadioButtons.

uint16_t size

The current number of added RadioButtons.

Callback< RadioButtonGroup, const AbstractButton &> radioButtonClicked

Callback that is attached to the RadioButtons.

• Callback< RadioButtonGroup, const AbstractButton &> radioButtonUnselected

Callback that is attached to the RadioButtons.

GenericCallback
 const AbstractButton &> * radioButtonSelectedCallback

The callback to be executed when a radio button belonging to this group is selected.

GenericCallback
 const AbstractButton &> * radioButtonDeselectedCallback

The callback to be executed when a radio button belonging to this group is deselected.

7.175.1 Detailed Description

```
\label{lem:condition} \begin{split} & template {<} uint16\_t \ CAPACITY{>} \\ & class \ touchgfx::RadioButtonGroup {<} \ CAPACITY{>} \end{split}
```

Class for handling a collection of RadioButtons. The RadioButtonGroup handles the de-selection of radio buttons when a new selection occurs. A callback is executed when a new selection occurs reporting the newly selected RadioButton.

Template class: specify a CAPACITY, that is the number of RadioButtons to store.

Template Parameters

CAPACITY Type of the capacity.	CAPACITY	Type of the capacity.
--------------------------------	----------	-----------------------

See also

RadioButton

7.175.2 Constructor & Destructor Documentation

7.175.2.1 RadioButtonGroup()

```
RadioButtonGroup ( ) [inline]
```

Default constructor.

7.175.2.2 ∼RadioButtonGroup()

```
\sim\!\!\text{RadioButtonGroup ()}\quad [\text{inline}]\text{, [virtual]}
```

Destructor.

7.175.3 Member Function Documentation

```
7.175.3.1 add()
```

Add the RadioButton to the RadioButtonGroup. Only add as many RadioButtons as the stated CAPACITY. Checked by an assert.

Parameters

in radioButton the RadioButton that is to be a	d.
--	----

7.175.3.2 getDeselectionEnabled()

```
bool getDeselectionEnabled ( ) const [inline], [virtual]
```

Gets the current deselectionEnabled state.

Returns

The current deselectionEnabled state.

7.175.3.3 getRadioButton()

Gets the RadioButton at the specified index.

Parameters

index the index of the RadioButton to return.

Returns

the RadioButton at the specified index. Returns 0 if illegal index.

7.175.3.4 getSelectedRadioButton()

```
RadioButton * getSelectedRadioButton ( ) const [inline], [virtual]
```

Gets the selected RadioButton.

Returns

a pointer to the selected RadioButton. Returns 0 if no RadioButton is selected.

7.175.3.5 getSelectedRadioButtonIndex()

```
int32_t getSelectedRadioButtonIndex ( ) const [inline], [virtual]
```

Gets the index of the selected RadioButton.

Returns

the index of the selected RadioButton. Returns -1 if no RadioButton is selected.

7.175.3.6 radioButtonClickedHandler()

Handles the event that a RadioButton has been selected. deselects all other RadioButtons.

Parameters

radioButton	the RadioButton that has been selected.
-------------	---

7.175.3.7 radioButtonDeselectedHandler()

Handles the event that a RadioButton has been deselected.

Parameters

radioButton	the RadioButton that has been deselected.
-------------	---

7.175.3.8 setDeselectionEnabled()

Sets whether or not it is possible to deselect RadioButtons by clicking them when they are selected.

Parameters

	deselectionEnabled	true if it should be possible to deselect by click.
--	--------------------	---

7.175.3.9 setRadioButtonDeselectedHandler()

Associates an action to be performed when a radio button belonging to this group transition from selected to unselected.

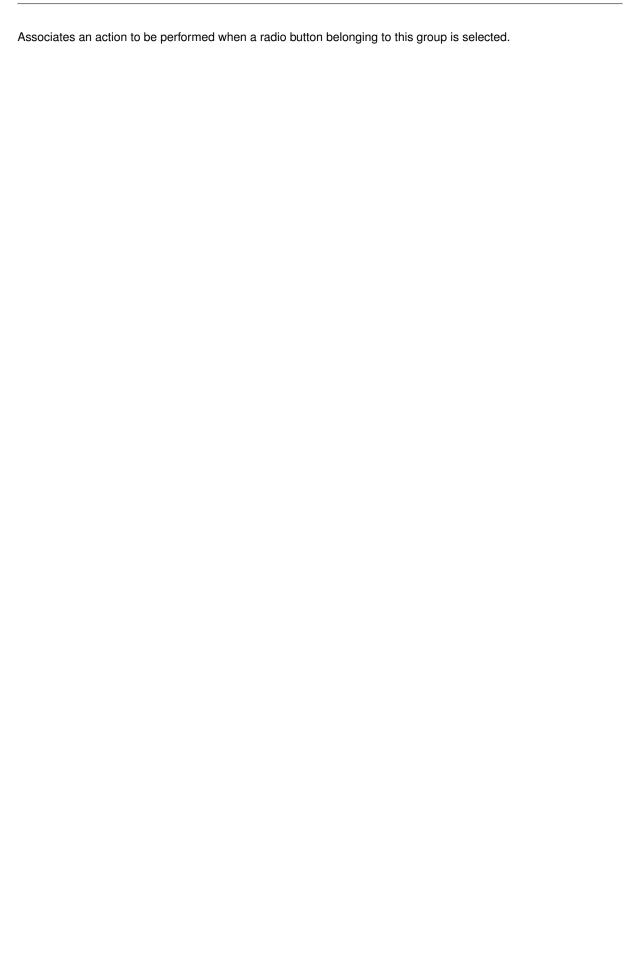
Parameters

callback	The callback to be executed. The callback will be given a reference to the RadioButton that was
	selected.

See also

GenericCallback

7.175.3.10 setRadioButtonSelectedHandler()



Parameters

callback	The callback to be executed. The callback will be given a reference to the RadioButton that was
	selected.

See also

GenericCallback

7.175.3.11 setSelected()

Sets the specified RadioButton to be selected and deselects all other. Do not call before all RadioButtons have been added to the RadioButtonGroup. Will call the radioButtonSelected callback.

Parameters

in <i>radioButtor</i>	the RadioButton to be selected.
-----------------------	---------------------------------

7.176 Rasterizer Class Reference

Polygon Rasterizer that is used to render filled polygons with high-quality Anti- Aliasing.

```
#include <touchgfx/canvas_widget_renderer/Rasterizer.hpp>
```

Public Types

enum { POLY_BASE_SHIFT = 5, POLY_BASE_SIZE = 1 << POLY_BASE_SHIFT, POLY_BASE_MASK = POLY_BASE_SIZE - 1 }

Determine the sub pixel accuracy, to be more precise, the number of bits of the fractional part of the coordinates.

 enum { AA_SHIFT = 8, AA_NUM = 1 << AA_SHIFT, AA_MASK = AA_NUM - 1, AA_2NUM = AA_NUM * 2, AA_2MASK = AA_2NUM - 1 }

Determine the area accuracy, to be more precise, the number of bits of the fractional part of the areas when calculating scanlines.

enum FillingRule { FILL_NON_ZERO, FILL_EVEN_ODD }

Values that represent filling rules.

Public Member Functions

• Rasterizer ()

Default constructor.

· void reset ()

Resets this object.

void setFillingRule (FillingRule fillingRule)

Sets the filling rule to be used when rendering the outline.

void moveTo (int x, int y)

Move to.

void lineTo (int x, int y)

Line to.

· unsigned calculateAlpha (int area) const

Calculates the alpha.

 template < class Renderer > bool render (Renderer &r)

Renders this object.

void setMaxRenderY (int y)

Sets maximum render y coordinate.

bool wasOutlineTooComplex ()

Determines if we the outline was too complex to draw completely.

7.176.1 Detailed Description

Polygon Rasterizer that is used to render filled polygons with high-quality Anti- Aliasing. Internally, by default, the class uses integer coordinates in format 24.8, i.e. 24 bits for integer part and 8 bits for fractional - see POLY_BA← SE_SHIFT. This class can be used in the following way:

- 1. setFillingRule(FillingRule fr) optional.
- 2. reset()
- 3. moveTo(x, y) / lineTo(x, y) make the polygon. One can create more than one contour, but each contour must consist of at least 3 vertices, i.e. moveTo(x1, y1); lineTo(x2, y2); lineTo(x3, y3); is the absolute minimum of vertices that define a triangle. The algorithm does not check either the number of vertices nor coincidence of their coordinates, but in the worst case it just won't draw anything. The order of the vertices (clockwise or counterclockwise) is important when using the non-zero filling rule (fill_non_zero). In this case the vertex order of all the contours must be the same if you want your intersecting polygons to be without "holes". You actually can use different vertices order. If the contours do not intersect each other the order is not important anyway. If they do, contours with the same vertex order will be rendered without "holes" while the intersecting contours with different orders will have "holes".

setFillingRule() can be called anytime before "sweeping".

7.176.2 Member Enumeration Documentation

7.176.2.1 anonymous enum

anonymous enum

Enumerator

POLY_BASE_SHIFT	Number of bits reserved for fraction part.
POLY_BASE_SIZE	The value to divide or multiply with to convert to / from this format.
POLY_BASE_MASK	The value used to mask the fraction.

7.176.2.2 anonymous enum

anonymous enum

Enumerator

AA_SHIFT	Number of bits reserved for fraction part when calculating the area.
AA_NUM	The value to divide or multiply with to convert to / from this format.
AA_MASK	The value used to mask the fraction.
AA_2NUM	Number of fraction bits when multiplying two area numbers.
AA_2MASK	Mask for fraction bits when multiplying two area numbers.

7.176.2.3 FillingRule

enum FillingRule

Values that represent filling rules.

Enumerator

FILL_NON_ZERO	Filling rule to fill anything inside the outmost border of the outline.
FILL_EVEN_ODD	Filling rule to fill using xor rule inside the outline.

7.176.3 Constructor & Destructor Documentation

7.176.3.1 Rasterizer()

```
Rasterizer ( ) [inline]
```

Default constructor.

7.176.4 Member Function Documentation

7.176.4.1 calculateAlpha()

Calculates the alpha.

Parameters

area	The area.

Returns

The calculated alpha.

7.176.4.2 lineTo()

Line to.

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.176.4.3 moveTo()

Move to.

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.176.4.4 render()

```
template< class Renderer > bool render ( Renderer & r ) [inline]
```

Renders this object.

Template Parameters

Renderer	Type of the renderer.

Parameters

in	r	The Renderer to process.
----	---	--------------------------

Returns

true there was enough memory available to draw the outline and render the graphics, false if there was insufficient memory and nothing was drawn.

7.176.4.5 reset()

```
void reset ( ) [inline]
```

Resets this object. Basically this is done by resetting the the Outline.

7.177 Rect Class Reference 753

7.176.4.6 setFillingRule()

Sets the filling rule to be used when rendering the outline.

Parameters

```
fillingRule The filling rule.
```

7.176.4.7 setMaxRenderY()

```
void setMaxRenderY ( \quad \text{int } y \text{ ) } \quad [\text{inline}]
```

Sets maximum render y coordinate. This is passed to the Outline to avoid registering any Cell that has a y coordinate less than zero of higher than the given y.

Parameters

y The max y coordinate to render for the Outline.

7.176.4.8 wasOutlineTooComplex()

```
bool wasOutlineTooComplex ( ) [inline]
```

Determines if we the outline was too complex to draw completely.

Returns

True if it was too complex, false if not.

7.177 Rect Class Reference

Class representing a Rectangle with a few convenient methods.

```
#include <touchgfx/hal/Types.hpp>
```

Public Member Functions

• Rect ()

Default constructor.

Rect (int16_t x, int16_t y, int16_t width, int16_t height)
 Constructor.

• int16_t right () const

Gets the x coordinate of the right edge of the Rect.

• int16_t bottom () const

Gets the y coordinate of the bottom edge of the Rect.

• bool intersect (int16_t otherX, int16_t otherY) const

Determines whether specified point lies inside this rectangle.

• bool intersect (const Rect &other) const

Determines whether specified rectangle intersects with this rectangle.

• bool includes (const Rect &other) const

Determines whether the specified rectangle is completely included in this rectangle.

• Rect operator& (const Rect &other) const

Gets a rectangle describing the intersecting area between this rectangle and the supplied rectangle.

void operator&= (const Rect &other)

Assigns this Rect to the intersection of the current Rect and the assigned Rect.

void expandToFit (const Rect &other)

Increases the area covered by this rectangle to encompass the area covered by supplied rectangle.

• bool operator== (const Rect &other) const

Compares equality of two Rect by the dimensions and position of these.

• bool operator!= (const Rect &other) const

Opposite of the == operator.

• bool isEmpty () const

Query if this object is empty.

• uint32_t area () const

Calculate the area of the rectangle.

Public Attributes

int16 t x

The x coordinate.

int16 t y

The y coordinate.

· int16_t width

The width.

· int16_t height

The height.

7.177.1 Detailed Description

Class representing a Rectangle with a few convenient methods. Size: 8 bytes.

7.177.2 Constructor & Destructor Documentation

```
7.177.2.1 Rect() [1/2]
```

Rect () [inline]

Default constructor. Resulting in an empty Rect with coordinates 0,0.

7.177 Rect Class Reference 755

7.177.2.2 Rect() [2/2]

Constructor.

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width.
height	The height.

7.177.3 Member Function Documentation

7.177.3.1 area()

```
uint32_t area ( ) const [inline]
```

Calculate the area of the rectangle.

Returns

area of the rectangle.

7.177.3.2 bottom()

```
int16_t bottom ( ) const [inline]
```

Gets the y coordinate of the bottom edge of the Rect.

Returns

y coordinate of the buttom edge.

7.177.3.3 expandToFit()

Increases the area covered by this rectangle to encompass the area covered by supplied rectangle.

Parameters

other	The other rectangle.

7.177.3.4 includes()

```
bool includes ( {\tt const\ Rect\ \&\ other\ )\ const\ [inline]}
```

Determines whether the specified rectangle is completely included in this rectangle.

Parameters

Returns

true if the specified rectangle is completely included.

7.177.3.5 intersect() [1/2]

Determines whether specified point lies inside this rectangle.

Parameters

otherX	The x coordinate of the point.
otherY	The y coordinate of the point.

Returns

true if point lies inside rectangle.

7.177.3.6 intersect() [2/2] bool intersect (

 ${\tt const\ Rect\ \&\ other\)\ const\ [inline]}$ Determines whether specified rectangle intersects with this rectangle.

Parameters

other	The other rectangle.

Returns

true if the two rectangles intersect.

7.177 Rect Class Reference 757

7.177.3.7 isEmpty()

```
bool isEmpty ( ) const [inline]
```

Query if this object is empty.

Returns

true if any of the dimensions are 0.

7.177.3.8 operator"!=()

Opposite of the == operator.

Parameters

other	The Rect to compare with.
-------	---------------------------

Returns

true if the compared Rect differ in dimensions or coordinates.

7.177.3.9 operator&()

Gets a rectangle describing the intersecting area between this rectangle and the supplied rectangle.

Parameters

other	The other rectangle.
-------	----------------------

Returns

Intersecting rectangle or Rect(0, 0, 0, 0) in case of no intersection.

7.177.3.10 operator&=()

Assigns this Rect to the intersection of the current Rect and the assigned Rect. The assignment will result in a Rect(0, 0, 0, 0) if they do not intersect.

Parameters

other	The rect to intersect with

7.177.3.11 operator==()

Compares equality of two Rect by the dimensions and position of these.

Parameters

other	The Rect to compare with.
-------	---------------------------

Returns

true if the compared Rect have the same dimensions and coordinates.

7.177.3.12 right()

```
int16_t right ( ) const [inline]
```

Gets the x coordinate of the right edge of the Rect.

Returns

x coordinate of the right edge.

7.178 Renderer Class Reference

This class template is used basically for rendering scan lines.

```
#include <touchgfx/canvas_widget_renderer/Renderer.hpp>
```

Public Member Functions

• Renderer ()

Default constructor.

• Renderer (RenderingBuffer &renderingBuffer, AbstractPainter &painter)

Constructor.

· void setRenderingBuffer (RenderingBuffer &renderingBuffer)

Sets rendering buffer.

void render (const Scanline &scanline)

Render the given Scanline in the given color.

• RenderingBuffer & getRenderingBuffer ()

Gets the getRenderingBuffer.

7.178.1 Detailed Description

This class template is used basically for rendering scanlines. The 'Span' argument is one of the span renderers, such as SpanRGB565 and others.

7.178.2 Constructor & Destructor Documentation

```
7.178.2.1 Renderer() [1/2]
Renderer ( ) [inline]
```

Default constructor. Function setRenderingBuffer() should be called to specify where the polygon should be rendered.

Constructor.

Parameters

in	renderingBuffer	The screen buffer to render the polygon in.
in	painter	The painter to use for drawing individual pixels in a scanline.

7.178.3 Member Function Documentation

7.178.3.1 getRenderingBuffer()

```
RenderingBuffer & getRenderingBuffer ( ) [inline]
```

Gets the getRenderingBuffer.

Returns

A RenderingBuffer&

Parameters

scanline	The Scanline

Render the given Scanline in the given color.

7.178.3.3 setRenderingBuffer()

Sets rendering buffer.

Parameters

in	renderingBuffer	The screen buffer to render the polygon in.
----	-----------------	---

7.179 RenderingBuffer Class Reference

Rendering buffer wrapper.

```
#include <touchgfx/canvas_widget_renderer/RenderingBuffer.hpp>
```

Public Member Functions

· RenderingBuffer ()

Default constructor.

∼RenderingBuffer ()

Destructor.

 RenderingBuffer (unsigned char *buf_, unsigned char xAdjust_, unsigned width_, unsigned height_, int stride)

Constructor.

- void attach (unsigned char *buf_, unsigned char xAdjust_, unsigned width_, unsigned height_, int stride_)
 Attaches a buffer.
- · unsigned char getXAdjust () const

Gets x coordinate adjust.

· unsigned getWidth () const

Gets the width.

unsigned getHeight () const

Gets the height.

bool inbox (int x, int y) const

Tests if a given coordinate is inside the RenderingBuffer.

unsigned char * row (unsigned y)

Gets a pointer to the given row in the RenderingBuffer.

const unsigned char * row (unsigned y) const

Gets a pointer to the given row in the RenderingBuffer.

7.179.1 Detailed Description

Rendering buffer wrapper. This class does not know anything about memory organizations, all it does it keeps an array of pointers to each pixel row. The general rules of rendering are as follows.

1. Allocate or create somehow a rendering buffer itself. Since the library does not depend on any particular platform or architecture it was decided that it's your responsibility to create and destroy rendering buffers properly. You can use any available mechanism to create it - you can use a system API function, simple memory allocation, or even statically defined array. You also should know the memory organization (or possible variants) in your system. For example, there's an R,G,B or B,G,R organizations with one byte per component (three bytes per pixel) is used very often. So, if you intend to use class render_bgr24, for example, you should allocate at least width*height*3 bytes of memory.

- 2. Create a RenderingBuffer object and then call method attach(). It requires a pointer to the buffer itself, width and height of the buffer in pixels, and the length of the row in bytes. All these values must properly correspond to the memory organization. The argument stride is used because in reality the row length in bytes does not obligatory correspond with the width of the image in pixels, i.e. it cannot be simply calculated as width_in_
 pixels * bytes_per_pixel. For example, it must be aligned to 4 bytes in Windows bitmaps. Method attach() can be called more than once. The execution time of it is very little, still it allocates memory of heigh * sizeof(char*) bytes and has a loop while (height—) {...}, so it's unreasonable to call it every time before drawing any single pixel:-)
- 3. Create an object (or a number of objects) of a rendering class, such as renderer_bgr24_solid, renderer bgr24_image and so on. These classes require a pointer to the RenderingBuffer object, but they do not perform any considerable operations except storing this pointer. So, rendering objects can be created on demand almost any time. These objects know about concrete memory organization (this knowledge is hard coded), so actually, the memory you allocated or created in clause 1 should actually be in correspondence to the needs of the rendering class.
- 4. Render your image using rendering classes, for example, Rasterizer
- 5. Display the result, or store it, or whatever. It's also your responsibility and depends on the platform.

7.179.2 Constructor & Destructor Documentation

Constructor.

Parameters

in	buf_	Pointer to the frame buffer where the image is rendered.
	x⇔ Adjust⇔ –	Horizontal adjustment of the x coordinate, used when bits per pixel is less than eight which implies that a uint8_t pointer cannot precisely address the start of the frame buffer.
	width_	The width of the frame buffer to write.
	height⇔ –	The height of the frame buffer to write.
	stride⊷	How much to add the a pointer inside the frame buffer to advance to the next line in the frame
	_	buffer.

7.179.3 Member Function Documentation

7.179.3.1 attach()

```
void attach (
          unsigned char * buf_,
          unsigned char xAdjust_,
          unsigned width_,
          unsigned height_,
          int stride_ )
```

Attaches a buffer. Can be used if the buffer is not ready when the Rendering buffer is created initially.

Parameters

in	buf_	Pointer to the frame buffer where the image is rendered.
	x⇔ Adjust⇔ –	Horizontal adjustment of the x coordinate, used when bits per pixel is less than eight which implies that a uint8_t pointer cannot precisely address the start of the frame buffer.
	width_	The width of the frame buffer to write.
	height↔ –	The height of the frame buffer to write.
	stride← -	How much to add the a pointer inside the frame buffer to advance to the next line in the frame buffer.

7.179.3.2 getHeight()

```
unsigned getHeight ( ) const [inline]
```

Gets the height.

Returns

The height.

7.179.3.3 getWidth()

```
unsigned getWidth ( ) const [inline]
```

Gets the width.

Returns

The width.

7.179.3.4 getXAdjust()

```
unsigned char getXAdjust ( ) const [inline]
```

Gets x coordinate adjust.

Returns

The x coordinate adjust.

7.179.3.5 inbox()

```
bool inbox (  \mbox{int } x, \\ \mbox{int } y \mbox{) const [inline]}
```

Tests if a given coordinate is inside the RenderingBuffer.

Parameters

Х	The x coordinate.
У	The y coordinate.

Returns

true if (x,y) is inside the RenderingBuffer, false otherwise.

```
7.179.3.6 row() [1/2] unsigned char * row ( unsigned y ) [inline]
```

Gets a pointer to the given row in the RenderingBuffer.

Parameters

```
y The line number, ie the row.
```

Returns

The pointer to the start of the given line in the RenderingBuffer.

Gets a pointer to the given row in the RenderingBuffer.

Parameters

```
y The line number, ie the row.
```

Returns

The pointer to the start of the given line in the RenderingBuffer.

7.180 RepeatButton Class Reference

A button with two states.

#include <touchgfx/widgets/RepeatButton.hpp>

Public Member Functions

• RepeatButton ()

Default constructor.

virtual void setDelay (int delay)

Sets the delay.

virtual int getDelay ()

Gets the delay.

• virtual void setInterval (int interval)

Sets the interval.

• virtual int getInterval ()

Gets the interval.

virtual void handleClickEvent (const touchgfx::ClickEvent &event)

Handles the click event.

virtual void handleTickEvent ()

Handles the tick event.

Additional Inherited Members

7.180.1 Detailed Description

A button consists of two images, one for its normal state and one when it is pressed down. The button activates its pressed action immediately, the after a given delay and then repeatedly after an interval.

See also

Button

7.180.2 Constructor & Destructor Documentation

7.180.2.1 RepeatButton()

RepeatButton ()

Default constructor. Sets delay to 10 ticks and interval to 5 ticks.

See also

setDelay setInterval

7.180.3 Member Function Documentation

```
7.180.3.1 getDelay()

int getDelay ( ) [virtual]

Gets the delay in ticks.

Returns

The delay.

See also

setDelay

7.180.3.2 getInterval()

int getInterval ( ) [virtual]

Gets the interval in ticks.
```

7.180.3.3 handleClickEvent()

The interval.

Handles the click event by immediately activating the button and then setting up a timer to repeatedly activate the button.

Parameters

Returns

```
event The event.
```

Reimplemented from AbstractButton.

7.180.3.4 handleTickEvent()

```
void handleTickEvent ( ) [virtual]
```

Handles the tick event that takes care of counting down until the next time the buttons should be activated.

Reimplemented from Drawable.

7.180.3.5 setDelay()

```
void setDelay ( int \ \textit{delay} \ ) \quad [virtual]
```

Sets the number of ticks from the first button activation until the next time it gets activated.

Parameters

```
delay The delay.
```

See also

setInterval getDelay

7.180.3.6 setInterval()

```
void setInterval (
                int interval ) [virtual]
```

Sets the interval in number of ticks between each each activation of the pressed button.

Parameters

interval	The interval.
----------	---------------

See also

setDelay getInterval

7.181 RepeatButtonTrigger Class Reference

A repeat button trigger.

#include <touchgfx/containers/buttons/RepeatButtonTrigger.hpp>

Public Member Functions

• RepeatButtonTrigger ()

Default constructor.

virtual ~RepeatButtonTrigger ()

Destructor.

void setDelay (int delay)

Sets a delay.

• int getDelay ()

Gets the delay.

void setInterval (int interval)

Sets an interval.

• int getInterval ()

Gets the interval.

void handleClickEvent (const touchgfx::ClickEvent &event)

Handles the click event described by event.

void handleTickEvent ()

Handles the tick event.

Additional Inherited Members

7.181.1 Detailed Description

A repeat button trigger. This trigger will create a button that reacts to a consistent touch. This means it will call the action repeatedly as long as it is touched.

The RepeatButtonTrigger can be combined with one or more of the ButtonStyle classes to create a functional button.

7.181.2 Member Function Documentation

```
7.181.2.1 getDelay()
int getDelay ( ) [inline]
Returns
    The delay.

7.181.2.2 getInterval()
```

int getInterval () [inline]

Returns

The interval.

7.181.2.3 handleClickEvent()

Parameters

event The event.

Reimplemented from Drawable.

7.181.2.4 setDelay()

Parameters

delay The delay.

7.181.2.5 setInterval()

```
void setInterval (
                int interval ) [inline]
```

Parameters

interval The interval.

7.182 ScalableImage Class Reference

Widget for representing a scaled version of a bitmap.

#include <touchgfx/widgets/ScalableImage.hpp>

Public Types

enum ScalingAlgorithm { NEAREST_NEIGHBOR, BILINEAR_INTERPOLATION }

Rendering algorithms of the scaled bitmap.

Public Member Functions

· ScalableImage ()

Default constructor.

virtual ∼ScalableImage ()

Destructor.

virtual void setBitmap (const Bitmap &bmp)

Sets the bitmap for the image.

• Bitmap getBitmap () const

Gets the bitmap for the image.

virtual void setAlpha (uint8_t alpha)

Sets the alpha channel for the image.

• virtual uint8_t getAlpha () const

Gets the current alpha value.

virtual void setScalingAlgorithm (ScalingAlgorithm algorithm)

Sets the algorithm to be used.

· virtual ScalingAlgorithm getScalingAlgorithm ()

Gets the algorithm used when rendering.

virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

· virtual Rect getSolidRect () const

Gets solid rectangle.

• virtual uint16_t getType () const

For GUI testing only.

Protected Member Functions

void drawTriangle (const Rect &invalidatedArea, uint16_t *fb, const float *triangleXs, const float *triangleYs, const float *triangleZs, const float *triangleUs, const float *triangleVs) const

Draw a triangle part of the bitmap.

RenderingVariant lookupRenderVariant () const

Looks up the appropriate render variant based on the bitmap format and scaling algorithm.

Protected Attributes

· ScalingAlgorithm currentScalingAlgorithm

The current scaling algorithm.

· Bitmap bitmap

The bitmap to render.

• uint8_t alpha

An alpha value that is applied to the entire image.

Additional Inherited Members

7.182.1 Detailed Description

Widget for representing a scaled version of a bitmap. Simply change the width/height of the widget to resize the image. The quality of the scaled image depends of the rendering algorithm used. The rendering algorithm can be changed dynamically. Please note that scaling images is done at runtime and requires a lot of calculations. Therefore use it with some care.

Note that this widget does not support 1 bit per pixel color depth.

See also

Widget

7.182.2 Member Enumeration Documentation

7.182.2.1 ScalingAlgorithm

```
enum ScalingAlgorithm
```

Rendering algorithms of the scaled bitmap.

NEAREST_NEIGHBOR: Fast but not a very good image quality. Good for fast animations.

BILINEAR_INTERPOLATION: Slow but good image quality. Good for static representation of a scaled image.

7.182.3 Constructor & Destructor Documentation

7.182.3.1 ScalableImage()

```
ScalableImage ( )
```

Default constructor.

7.182.3.2 \sim ScalableImage()

```
\simScalableImage ( ) [virtual]
```

Destructor.

7.182.4 Member Function Documentation

```
7.182.4.1 draw()
void draw (
```

Draws the given invalidated area.

Parameters

invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.
-----------------	--

const Rect & invalidatedArea) const [virtual]

See also

Drawable::draw()

Implements Drawable.

7.182.4.2 drawTriangle()

Draw a triangle part of the bitmap.

Parameters

	invalidatedArea	The invalidated area.
in,out	fb	If non-null, the fb.
	triangleXs	The triangle xs.
	triangleYs	The triangle ys.
	triangleZs	The triangle zs.
	triangleUs	The triangle us.
	triangleVs	The triangle vs.

```
7.182.4.3 getAlpha()
uint8_t getAlpha ( ) const [inline], [virtual]
Gets the current alpha value.
Returns
     The current alpha value.
7.182.4.4 getBitmap()
Bitmap getBitmap ( ) const [inline]
Gets the bitmap for the image.
Returns
     the small bitmap.
7.182.4.5 getScalingAlgorithm()
ScalingAlgorithm getScalingAlgorithm ( ) [virtual]
Gets the algorithm used when rendering.
Returns
     The algorithm used when rendering.
7.182.4.6 getSolidRect()
Rect getSolidRect ( ) const [virtual]
Gets solid rectangle.
Returns
     largest possible solid rect.
See also
     Drawable::getSolidRect()
Implements Drawable.
7.182.4.7 getType()
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_SCALABLEIMAGE.

Reimplemented from Widget.

7.182.4.8 lookupRenderVariant()

```
RenderingVariant lookupRenderVariant ( ) const [protected]
```

Looks up the appropriate render variant based on the bitmap format and scaling algorithm.

Returns

A Rendering Variant.

7.182.4.9 setAlpha()

Sets the alpha channel for the image.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.182.4.10 setBitmap()

```
void setBitmap ( {\tt const\ Bitmap\ \&\ \it bmp\ )} \quad [{\tt virtual}]
```

Sets the bitmap for the image.

Parameters

bmp	The bitmap to be used by the widget.

7.182.4.11 setScalingAlgorithm()

Sets the algorithm to be used.

Parameters

algorithm	The algorithm to use when rendering.

7.183 Scanline Class Reference

This class is used to transfer data from class Outline (or a similar one) to the rendering buffer.

#include <touchgfx/canvas_widget_renderer/Scanline.hpp>

Classes

· class iterator

An iterator to help go through all the elements that make up a Scanline.

Public Member Functions

· Scanline ()

Default constructor.

virtual ∼Scanline ()

Destructor.

· void reset ()

Resets the Scanline object in preparation for the handling the next Scanline.

void resetSpans ()

Resets the spans in preparation for the next Scanline.

void addCell (int x, int y, unsigned cover)

Adds a single cell to the current Scanline.

void addSpan (int x, int y, unsigned len, unsigned cover)

Adds a span of cells to the current Scanline.

• int isReady (int y) const

Checks if a Scanline is ready for rendering.

• int getY () const

Gets y coordinate, i.e. the vertical offset of the Scanline.

• unsigned getNumSpans () const

Gets number spans in the Scanline.

7.183.1 Detailed Description

This class is used to transfer data from class Outline (or a similar one) to the rendering buffer. It's organized very simple. The class stores information of horizontal spans to render it into a pixel-map buffer. Each span has initial X, length, and an array of bytes that determine the alpha values for each pixel. So, the restriction of using this class is 256 levels of Anti-Aliasing, which is quite enough for any practical purpose. Before using this class you should know the minimal and maximal pixel coordinates of your scanline. The protocol of using is: 1. reset()

- 1. addCell() / addSpan() accumulate scanline. You pass y coordinate into these functions in order to make scanline know the last Y. Before calling addCell() / addSpan() you should check with method isReady(y) if the last Y has changed. It also checks if the scanline is not empty. When forming one scanline the next x coordinate must be always greater than the last stored one, i.e. it works only with ordered coordinates.
- 2. If the current scanline isReady() you should render it and then call resetSpans() before adding new cells/spans.
- 3. Rendering:

Scanline provides an iterator class that allows you to extract the spans and the cover values for each pixel. Be aware that clipping has not been done yet, so you

should perform it yourself. Use Scanline::iterator to render spans:

int base X = scanline.getBase(X); // base X. Should be added to the span's X // "scanline" is a const reference to the // scanline passed in.

int y = scanline.y(); // y coordinate of the scanline

...Perform vertical clipping here...

Scanline::iterator span(scanline);

unsigned char* row = renderingBuffer->row(y); // The the address of the beginning // of the current row

unsigned num_spans = scanline.getNumSpans(); // Number of spans. It's guaranteed that // numSpans is always greater than 0.

```
do { int x = \text{span.next}() + \text{base}X; // The beginning X of the span}
```

const int8u covers* = span.getCovers(); // The array of the cover values

int numPix = span.getNumPix(); // Number of pixels of the span. // Always greater than 0, still we // should use "int" instead of // "unsigned" because it's more // convenient for clipping

...Perform horizontal clipping here... ...you have x, covers, and pix_Fromcount...

unsigned char* dst = row + x; // Calculate the start address of the row. // In this case we assume a simple // grayscale image 1-byte per pixel. do { *dst++ = *covers++; // Hypotetical rendering. } while (-numPix); } while (-numSpans); // numSpans cannot be 0, so this loop is quite safe

The question is: why should we accumulate the whole scanline when we could render just separate spans when they're ready? That's because using the scanline is in general faster. When is consists of more than one span the conditions for the processor cash system are better, because switching between two different areas of memory (that can be large ones) occurs less frequently.

7.183.2 Constructor & Destructor Documentation

```
7.183.2.1 Scanline()
```

```
Scanline ( )
```

Default constructor. Initiate a Scanline by setting up pointers to store covers, and counts.

```
7.183.2.2 ∼Scanline()
```

```
~Scanline ( ) [inline], [virtual]
```

Destructor.

7.183.3 Member Function Documentation

7.183.3.1 addCell()

```
\begin{tabular}{ll} FORCE\_INLINE\_FUNCTION & void & addCell & ( & int $x$, & \\ & int $y$, & \\ & unsigned & cover & ( & ) \\ \end{tabular}
```

Adds a single cell to the current Scanline. Works just like invoking addSpan() with a len=1.

Parameters

X	The x coordinate.
У	The y coordinate.
cover	The cover.

7.183.3.2 addSpan()

```
void addSpan (
    int x,
    int y,
    unsigned len,
    unsigned cover )
```

Adds a span of cells to the current Scanline. Works like calling addCell() len times.

Parameters

X	The x coordinate.
У	The y coordinate.
len	The length.
cover	The cover.

7.183.3.3 getNumSpans()

```
unsigned getNumSpans ( ) const [inline]
```

Gets number spans in the Scanline.

Returns

The number spans.

7.183.3.4 getY()

```
int getY ( ) const [inline]
```

Gets y coordinate, i.e. the vertical offset of the Scanline. This allows easy positioning of the Outline. The y coordinate is setup through function reset().

Returns

The y coordinate.

7.183.3.5 isReady()

```
\label{force_interpolation} \mbox{FORCE\_INLINE\_FUNCTION int isReady (} \\ \mbox{int } y \mbox{ ) const}
```

Checks if a Scanline is ready for rendering. A Scanline is ready for rendering when the y coordinate has changed. Since all the cells are sorted, a change in the y coordinate means that we have moved to the next Scanline and thus the collected data for the Scanline must be rendered before we register cells for the next Scanline.

Parameters

```
y The y coordinate.
```

Returns

True if the given y coordinate differs from the y coordinate for the cells in the current Scanline.

7.183.3.6 reset()

void reset ()

Resets the Scanline object in preparation for the handling the next Scanline.

7.183.3.7 resetSpans()

```
FORCE_INLINE_FUNCTION void resetSpans ( )
```

Resets the spans in preparation for the next Scanline. Identical to calling reset() without changing the dx_ and dy_ parameters from the previous call to reset().

7.184 Screen Class Reference

A Screen represents a full-screen drawable area. Applications create specific screens by subclassing this class.

```
#include <touchgfx/Screen.hpp>
```

Public Member Functions

• Screen ()

Default constructor.

virtual ∼Screen ()

Destructor.

• void draw ()

Tells the screen to draw its entire area.

void startSMOC (const Rect &invalidatedArea)

Starts a JSMOC run, analyzing what parts of what widgets should be redrawn.

void JSMOC (const Rect &invalidatedArea, Drawable *widgetToDraw)

Recursive JSMOC function. This is the actual occlusion culling implementation.

virtual void draw (Rect &rect)

Tell the screen to draw the specified area.

• virtual void setupScreen ()

Called by Application::switchScreen() when this screen is going to be displayed.

• virtual void afterTransition ()

Called by Application::handleTick() when the transition to the screen is done.

virtual void tearDownScreen ()

Called by Application::switchScreen() when this screen will no longer be displayed.

virtual void handleClickEvent (const ClickEvent &evt)

Traverse the drawables in reverse z-order and notify them of a click event.

virtual void handleDragEvent (const DragEvent &evt)

Traverse the drawables in reverse z-order and notify them of a drag event.

virtual void handleGestureEvent (const GestureEvent &evt)

Handle gestures. Traverses drawables in reverse-z and notifies them of the gesture.

virtual void handleTickEvent ()

Called by the Application on the current screen with a frequency of Application::TICK_INTERVAL_MS.

virtual void handleKeyEvent (uint8 t key)

Called by the Application on the reception of a "key", the meaning of which is platform/application specific.

• bool usingSMOC () const

Determines if using JSMOC.

· void bindTransition (Transition &trans)

Enables the transition to access the containers.

• Container & getRootContainer ()

Obtain a reference to the root container of this screen.

Protected Member Functions

void useSMOCDrawing (bool enabled)

Determines whether to use JSMOC or painter's algorithm for drawing.

void add (Drawable &d)

Add a drawable to the content container.

void remove (Drawable &d)

Removes a drawable from the content container.

Protected Attributes

Container container

The container contains the contents of the screen.

Drawable * focus

The drawable currently in focus (set when DOWN_PRESSED is received).

7.184.1 Detailed Description

A Screen represents a full-screen drawable area. Applications create specific screens by subclassing this class.

Each screen has a root container to which drawables can be added.

This class makes sure to delegate draw requests and various events to the appropriate drawables in correct order.

7.184.2 Constructor & Destructor Documentation

7.184.2.1 Screen()

Screen ()

Default constructor.

```
7.184.2.2 \simScreen()
```

```
~Screen ( ) [inline], [virtual]
```

Destructor.

7.184.3 Member Function Documentation

```
7.184.3.1 add()  \begin{tabular}{ll} \begin{tabular}{ll} void & add & ( & & \\ & & Drawable & d & ) & [inline], & [protected] \end{tabular}
```

Add a drawable to the content container.

Note

Must not be called with a Drawable that was already added to the screen. If in doubt, call remove() first.

Parameters

in	d	The Drawable to add.
----	---	----------------------

7.184.3.2 afterTransition()

```
void afterTransition ( ) [inline], [virtual]
```

Called by Application::handleTick() when the transition to the screen is done. Base version does nothing, but override to do screen specific initialization code that has to be done after the transition to the screen.

See also

touchgfx::Application::handleTick()

7.184.3.3 bindTransition()

Enables the transition to access the containers.

Parameters

in	trans	The transition to bind.

```
7.184.3.4 draw() [1/2]
```

```
void draw ( )
```

Tells the screen to draw its entire area.

Note

The more specific draw(Rect&) version is preferred when possible.

Tell the screen to draw the specified area. Will traverse the drawables tree in z- order and delegate draw to them.

Note

The given rect must be in absolute coordinates.

Parameters

```
in rect The area in absolute coordinates.
```

7.184.3.6 getRootContainer()

```
Container & getRootContainer ( ) [inline]
```

Obtain a reference to the root container of this screen.

Returns

The root container.

7.184.3.7 handleClickEvent()

Traverse the drawables in reverse z-order and notify them of a click event.

Parameters

```
evt The event to handle.
```

7.184.3.8 handleDragEvent()

Traverse the drawables in reverse z-order and notify them of a drag event.

Parameters

```
evt The event to handle.
```

7.184.3.9 handleGestureEvent()

```
void handleGestureEvent ( {\tt const~GestureEvent~\&~evt~)} \quad [{\tt virtual}]
```

Handle gestures. Traverses drawables in reverse-z and notifies them of the gesture.

Parameters

```
evt The event to handle.
```

7.184.3.10 handleKeyEvent()

Called by the Application on the reception of a "key", the meaning of which is platform/application specific. Default implementation does nothing.

Parameters

```
key The key to handle.
```

7.184.3.11 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

Called by the Application on the current screen with a frequency of Application::TICK_INTERVAL_MS.

7.184.3.12 JSMOC()

Recursive JSMOC function. This is the actual occlusion culling implementation.

Parameters

	in	invalidatedArea	The area to redraw, expressed in absolute coordinates.
ſ	in	widgetToDraw	Widget currently being drawn.

7.184.3.13 remove()

Removes a drawable from the content container. Safe to call even if the drawable was never added (in which case nothing happens).

Parameters

in	d	The Drawable to remove.
T11	u	THE DIAWADIC to remove

7.184.3.14 setupScreen()

```
void setupScreen ( ) [inline], [virtual]
```

Called by Application::switchScreen() when this screen is going to be displayed. Base version does nothing, but place any screen specific initialization code in an overridden version.

See also

touchgfx::Application::switchScreen()

7.184.3.15 startSMOC()

Starts a JSMOC run, analyzing what parts of what widgets should be redrawn.

Parameters

in	invalidatedArea	The area to redraw, expressed in absolute coordinates.

7.184.3.16 tearDownScreen()

```
void tearDownScreen ( ) [inline], [virtual]
```

Called by Application::switchScreen() when this screen will no longer be displayed. Base version does nothing, but place any screen specific cleanup code in an overridden version.

See also

touchgfx::Application::switchScreen()

7.184.3.17 useSMOCDrawing()

Determines whether to use JSMOC or painter's algorithm for drawing.

Parameters

enabled

true if JSMOC should be enabled, false if disabled (meaning painter's algorithm is employed instead).

7.184.3.18 usingSMOC()

```
bool usingSMOC ( ) const [inline]
```

Returns

true if this screen uses the JSMOC drawing algorithm.

7.185 ScrollableContainer Class Reference

A ScrollableContainer is a container that allows its contents to be scrolled.

#include <touchgfx/containers/ScrollableContainer.hpp>

Public Member Functions

• ScrollableContainer ()

Default constructor.

virtual ∼ScrollableContainer ()

Destructor.

void enableHorizontalScroll (bool enable)

Enables horizontal scrolling.

• void enableVerticalScroll (bool enable)

Enables the vertical scroll.

virtual void isScrollableXY (bool &scrollX, bool &scrollY)

Is the ClickableContainer scrollable in either direction?

• void setScrollbarsVisible (bool newVisible)

Sets the visibility of the scrollbars, when the scrollable area is pressed.

• void setScrollbarsPermanentlyVisible ()

sets the visibility for the scrollbars to be permanent.

• virtual void add (Drawable &d)

Adds a Drawable instance as child to this ScrollableContainer.

virtual void getLastChild (int16_t x, int16_t y, Drawable **last)

Gets the last child in the container.

virtual void handleClickEvent (const ClickEvent &evt)

Handle the click event.

virtual void handleDragEvent (const DragEvent &evt)

Handle the drag event.

· virtual void handleGestureEvent (const GestureEvent &evt)

Gestures generate a scroll animation so these are intercepted in the same manner as drag events.

virtual void handleTickEvent ()

Handle tick events.

virtual Rect getContainedArea () const

Gets contained area.

· virtual void childGeometryChanged ()

Used to signal that the size of one or more children have changed.

· void reset ()

Resets the x/y coordinates of children.

virtual void moveChildrenRelative (int16_t deltaX, int16_t deltaY)

Moves the scrollable contents relatively.

void setMaxVelocity (uint16_t max)

Sets the maximum velocity of a scroll due to a swipe.

void setScrollThreshold (int16_t t)

Change the threshold which the first drag event received must exceed before initiating a scroll.

void setScrollbarsColor (colortype color)

Sets the color of the scroll bars.

void setScrollbarsAlpha (uint8 t alpha)

Sets the alpha value for the scroll bars.

void setScrollbarPadding (uint8_t padding)

Sets the amount of space the scrollbar has to its borders.

void setScrollbarWidth (uint8 t width)

Sets the width of the scrollbar.

int16_t getScrolledX () const

Gets the distance scrolled for the x-axis.

• int16 t getScrolledY () const

Gets the distance scrolled for the y-axis.

virtual uint16_t getType () const

For GUI testing only.

Protected Member Functions

· Rect getXScrollbar () const

Gets x coordinate of the scrollbar.

· Rect getYScrollbar () const

Gets y coordinate of the scrollbar.

Rect getXBorder (const Rect &xBar, const Rect &yBar) const

Gets the area where the horizontal scrollbar can move.

Rect getYBorder (const Rect &xBar, const Rect &yBar) const

Gets the area where the vertical scrollbar can move.

· void invalidateScrollbars ()

Invalidate the scrollbars.

virtual bool doScroll (int16_t deltaX, int16_t deltaY)

Method to actually scroll the container.

Protected Attributes

· uint8 t scrollbarPadding

The amount of padding. The scrollbar will have a bit of space to the borders of the container.

uint8_t scrollbarWidth

The width of the scrollbar.

· uint8_t scrollbarAlpha

The scrollbar is semitransparent.

colortype scrollbarColor

The color of the scrollbar.

uint16_t maxVelocity

The maximum velocity of a scroll (due to a swipe)

GestureEvent::GestureType accelDirection

The current direction (horizontal or vertical) of scroll.

Box xSlider

The horizontal scrollbar drawable.

Box ySlider

The vertical scrollbar drawable.

Drawable * pressedDrawable

The drawable child of this container which received the last ClickEvent::PRESSED notification. When scrolling, send this drawable a CANCEL event if the new x/y coords no longer matches this drawable.

• Drawable * lastDraggableChild

The drawable child of this container which should receive drag events. Note that only drag events in directions which cannot be scrolled by this ScrollableContainer will be forwarded to children.

• int16 t scrolledXDistance

The scrolled horizontal distance.

• int16_t scrolledYDistance

The scrolled vertical distance.

· int16 t scrollThreshold

The threshold which the first drag event received must exceed before scrolling. Default is 5.

int16 t pressedX

The x coordinate where the last ClickEvent::PRESSED was received.

int16_t pressedY

The y coordinate where the last ClickEvent::PRESSED was received.

bool isPressed

Is the container currently pressed (maybe show scrollbars)

bool isScrolling

Is the container scrolling (i.e. has overcome the initial larger drag that is required to initiate a scroll).

bool scrollableX

Is the container scrollable in the horizontal direction.

· bool scrollableY

Is the container scrollable in the vertical direction.

· bool scrollbarsVisible

Are scrollbars always visible.

· bool scrollbarsPermanentlyVisible

Are scrollbars alway visible.

· uint16 t scrollDuration

Number of ticks the scroll animation should use.

• int16_t beginningValue

Initial X or Y for calculated values in scroll animation.

int16 t targetValue

Target X or Y value for scroll animation.

uint16_t animationCounter

Current step/tick in scroll animation.

· bool animate

Is scroll animation currently active.

int16_t fingerAdjustmentX

How much should the finger be adjusted horizontally.

· int16_t fingerAdjustmentY

and how much vertically

· bool hasIssuedCancelEvent

true if the pressed drawable has received cancel event

Static Protected Attributes

static const uint8_t SCROLLBAR_LINE = 0

The scrollbar line.

static const uint16 t SCROLLBAR MIN VELOCITY = 5

The minimum velocity of a scroll due to a swipe.

static const uint16_t SCROLLBAR_MAX_VELOCITY = 17

The (default) maximum velocity of a scroll due to a swipe.

Additional Inherited Members

7.185.1 Detailed Description

A ScrollableContainer is a container that allows its contents to be scrolled. It will intercept drag operations and move child nodes accordingly.

The size of the ScrollableContainer should be the visible view port area. If the container contains drawables that are larger than the ScrollableContainer itself, scrolling is enabled.

Note

The ScrollableContainer will consume all DragEvents in the area covered by the container, and use.

See also

Container

7.185.2 Constructor & Destructor Documentation

```
7.185.2.1 ScrollableContainer()
```

```
ScrollableContainer ( )
```

Default constructor.

```
7.185.2.2 ∼ScrollableContainer()
```

```
\simScrollableContainer ( ) [inline], [virtual]
```

Destructor.

7.185.3 Member Function Documentation

```
7.185.3.1 add() \label{eq:condition} \mbox{void add (} \mbox{$\tt Drawable \& $d$ }) \mbox{ [virtual]}
```

Adds a Drawable instance as child to this ScrollableContainer.

Parameters

in d	The drawable.
-------------	---------------

Reimplemented from Container.

7.185.3.2 childGeometryChanged()

```
void childGeometryChanged ( ) [virtual]
```

This function can be called on parent nodes to signal that the size of one or more of its children have changed. Currently only used in ScrollableContainer to redraw scrollbars when the size of the scrolling contents changes.

See also

Drawable::childGeometryChanged

Reimplemented from Drawable.

7.185.3.3 doScroll()

Method to actually scroll the container. Passing negative values will scroll the items in the ScrollableContainer up / left, whereas positive values will scroll items down / right.

If the distance is larger than allowed, the deltas are adjusted down to make sure the contained items stay inside view.

Parameters

deltaX	The horizontal amount to scroll.
deltaY	The vertical amount to scroll.

Returns

did the container actually scroll. The call doScroll(0,0) will always return false.

7.185.3.4 enableHorizontalScroll()

```
void enableHorizontalScroll (
          bool enable ) [inline]
```

By default, scrolling in either direction is enabled, provided that the content is larger than the size of the scrollable container. This function can be used to explicitly (dis)allow scrolling in the horizontal direction, even if the content is larger than the container.

Parameters

enable	If true (default), horizontal scrolling is enabled. If false, scrolling is disabled.
--------	--

7.185.3.5 enableVerticalScroll()

```
void enableVerticalScroll (
          bool enable ) [inline]
```

Enables the vertical scroll. By default, scrolling in either direction is enabled, provided that the content is larger than the size of the scrollable container. This function can be used to explicitly (dis)allow scrolling in the vertical direction, even if the content is larger than the container.

Parameters

enable If true (default), vertical scrolling is enabled. If false, scrolling is disabled.

7.185.3.6 getContainedArea()

```
Rect getContainedArea ( ) const [virtual]
```

Gets contained area.

Returns

The contained area.

Reimplemented from Container.

7.185.3.7 getLastChild()

```
void getLastChild (
    int16_t x,
    int16_t y,
    Drawable ** last ) [inline], [virtual]
```

Gets the last child in the container. The ScrollableContainer needs to intercept click events, since the scrollbars are displayed upon reception of a PRESSED ClickEvent. The ScrollableContainer will automatically re-delegate the event to the appropriate child.

Parameters

		X	The x coordinate of the (click) event.	
Ī		У	The y coordinate of the (click) event.	
Ī	out	last	The last child intersecting x,y. ScrollableContainer intercepts these, so returns it self.	

Reimplemented from Container.

7.185.3.8 getScrolledX()

```
int16_t getScrolledX ( ) const
```

Gets the distance scrolled for the x-axis.

Returns

the distance scrolled for the x-axis.

7.185.3.9 getScrolledY()

```
int16_t getScrolledY ( ) const
```

Gets the distance scrolled for the y-axis.

Returns

the distance scrolled for the y-axis.

7.185.3.10 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_SCROLLABLECONTAINER.

Reimplemented from Container.

7.185.3.11 getXBorder()

Gets the area where the horizontal scrollbar can move.

Parameters

xBar	The current horizontal scrollbar, supplied for caching reasons.
yBar	The current vertical scrollbar, supplied for caching reasons.

Returns

The area.

7.185.3.12 getXScrollbar()

```
Rect getXScrollbar ( ) const [protected]
```

Gets x coordinate of the scrollbar.

Returns

The horizontal scrollbar area.

7.185.3.13 getYBorder()

Gets the area where the vertical scrollbar can move.

Parameters

xBar	The current horizontal scrollbar, supplied for caching reasons.
yBar	The current vertical scrollbar, supplied for caching reasons.

Returns

The area.

7.185.3.14 getYScrollbar()

```
Rect getYScrollbar ( ) const [protected]
```

Gets y coordinate of the scrollbar.

Returns

The vertical scrollbar area.

7.185.3.15 handleClickEvent()

```
void handleClickEvent (
                      const ClickEvent & evt ) [virtual]
```

Handle the click event. Get ready for scrolling, display scrollbars, etc. Send the click to appropriate child widget.

Parameters

evt	The ClickEvent.

Reimplemented from Drawable.

7.185.3.16 handleDragEvent()

Handle the drag event. Initiate a scrolling of the container. Update scrollbars.

Parameters

```
evt The DragEvent.
```

Reimplemented from Drawable.

7.185.3.17 handleGestureEvent()

```
void handleGestureEvent ( {\tt const~GestureEvent~\&~evt~)} \quad [{\tt virtual}]
```

Gestures generate a scroll animation so these are intercepted in the same manner as drag events.

Parameters

```
evt The GestureEvent.
```

Reimplemented from Drawable.

7.185.3.18 handleTickEvent()

```
void handleTickEvent ( ) [virtual]
```

Handle tick events. Used in updating the animation of the scroll.

Reimplemented from Drawable.

7.185.3.19 invalidateScrollbars()

```
void invalidateScrollbars ( ) [protected]
```

Invalidate the scrollbars.

7.185.3.20 isScrollableXY()

```
void isScrollableXY (
                bool & scrollX,
                bool & scrollY ) [inline], [virtual]
```

Is the ClickableContainer scrollable in either direction? Takes the width of the contained elements into account.

Parameters

in,out	scrollX	Is the container able to scroll horizontally.
in,out	scrollY	Is the container able to scroll vertically.

7.185.3.21 moveChildrenRelative()

```
void moveChildrenRelative (
          int16_t deltaX,
          int16_t deltaY ) [virtual]
```

Moves the scrollable contents relatively.

Parameters

deltaX	Horizontal displacement.
deltaY	Vertical displacement.

Reimplemented from Container.

7.185.3.22 reset()

```
void reset ( )
```

Resets the x/y coordinates of childrento the position they were in before the first drag event was received or to the position they were in the last time reset() was invoked.

7.185.3.23 setMaxVelocity()

```
void setMaxVelocity (
          uint16_t max ) [inline]
```

Sets the maximum velocity of a scroll due to a swipe.

Parameters

max	The maximum velocity of the scroll.
-----	-------------------------------------

7.185.3.24 setScrollbarPadding()

Sets the amount of space the scrollbar has to its borders.

Parameters

padding	The padding.
---------	--------------

7.185.3.25 setScrollbarsAlpha()

Sets the alpha value for the scroll bars.

Parameters

alpha	The alpha value. 255 = completely solid.
-------	--

7.185.3.26 setScrollbarsColor()

Sets the color of the scroll bars.

Parameters

7.185.3.27 setScrollbarsPermanentlyVisible()

```
void setScrollbarsPermanentlyVisible ( )
```

sets the visibility for the scrollbars to be permanent.

7.185.3.28 setScrollbarsVisible()

Sets the visibility of the scrollbars, when the scrollable area is pressed.

Parameters

newVisible	If true (default), the scrollbars are visible when scrollable area is pressed. If false, scrollbars are
	always hidden.

7.185.3.29 setScrollbarWidth()

Sets the width of the scrollbar.

Parameters

7.185.3.30 setScrollThreshold()

```
\verb"void setScrollThreshold" (
```

```
int16_t t ) [inline]
```

Change the threshold which the first drag event received must exceed before initiating a scroll.

Note

All subsequent scrolls will be processed regardless of threshold value until a ClickEvent::RELEASED is received.

Parameters

t The new threshold value.

7.186 ScrollBase Class Reference

A scroll base class.

#include <touchgfx/containers/scrollers/ScrollBase.hpp>

Public Member Functions

• ScrollBase ()

Default constructor.

virtual ∼ScrollBase ()

Destructor.

virtual void setWidth (int16_t width)

Sets width of the ScrollBase.

virtual void setHeight (int16_t height)

Sets height of the ScrollBase.

virtual void setHorizontal (bool horizontal)

Sets a horizontal layout.

· virtual bool getHorizontal () const

Gets the orientation of the drawables.

virtual void setCircular (bool circular)

Sets whether the list is circular or not.

virtual bool getCircular () const

Gets the circular setting.

• void setDrawableSize (int16_t drawableSize, int16_t drawableMargin)

Sets drawables size.

virtual int16_t getDrawableSize () const

Gets drawable size.

• virtual int16_t getDrawableMargin () const

Gets drawable margin.

virtual void setNumberOfItems (int16_t numberOfItems)

Sets number of items in the DrawableList.

• virtual int16_t getNumberOfItems () const

Gets number of items in the DrawableList.

void setEasingEquation (EasingEquation equation)

Sets easing equation.

void setAnimationSteps (int16 t steps)

Sets animation steps.

• uint16_t getAnimationSteps () const

Gets animation steps.

void setSwipeAcceleration (uint16_t acceleration)

Sets swipe acceleration.

• uint16_t getSwipeAcceleration () const

Gets swipe acceleration.

void setMaxSwipeItems (uint16 t maxItems)

Sets maximum swipe items.

• uint16_t getMaxSwipeItems () const

Gets maximum swipe items.

void setDragAcceleration (uint16_t acceleration)

Sets drag acceleration.

uint16_t getDragAcceleration () const

Gets drag acceleration.

• void allowHorizontalDrag (bool enable)

Enables horizontal scrolling.

• void allowVerticalDrag (bool enable)

Enables the vertical scroll.

virtual void animateToltem (int16 t itemIndex, int16 t animationSteps=-1)

Go to item.

void setItemSelectedCallback (GenericCallback< int16 t > &callback)

Sets Callback which will be called when the selected item is clicked.

void setAnimationEndedCallback (GenericCallback<> &callback)

Callback, called when the set animation ended.

void setItemPressedCallback (GenericCallback< int16_t > &callback)

Set Callback which will be called when a item is pressed.

• bool isAnimating () const

Query if this object is animating.

• void stopAnimation ()

Stops an animation.

virtual void handleDragEvent (const DragEvent &evt)

Handles the drag event described by evt.

· virtual void handleGestureEvent (const GestureEvent &evt)

Handles the gesture event described by evt.

virtual void handleTickEvent ()

Handles the tick event.

• virtual void itemChanged (int itemIndex)

Item changed.

virtual void initialize ()

Initializes the contents of all drawables.

Protected Types

• enum AnimationState { NO ANIMATION, ANIMATING GESTURE, ANIMATING DRAG }

Values that represent animation states.

Protected Member Functions

virtual void setOffset (int32 t offset)

Sets display offset of first item.

virtual int32_t getOffset () const

Gets display offset of first item.

virtual int32_t getPositionForItem (int16_t itemIndex)=0

Gets position for an item.

• int getNormalizedOffset (int offset) const

Gets normalized offset from a given offset.

virtual int32_t keepOffsetInsideLimits (int32_t newOffset, int16_t overShoot) const =0

Keep offset inside limits.

• virtual int32_t getNearestAlignedOffset (int32_t offset) const

Gets nearest offset aligned to a multiple of itemSize.

virtual void animateToPosition (int32_t position, int16_t steps=-1)

Animate to a new position/offset using the given number of steps.

Protected Attributes

· DrawableList list

The list.

• int16_t numberOfDrawables

Number of drawables.

• int16_t distanceBeforeAlignedItem

The distance before aligned item.

• int16 titemSize

Size of the item.

• uint16_t swipeAcceleration

The swipe acceleration.

· uint16_t dragAcceleration

The drag acceleration.

uint16_t maxSwipeItems

The maximum swipe items.

EasingEquation easingEquation

The easing equation.

• uint16_t defaultAnimationSteps

The animation steps.

• GenericCallback< int16_t > * itemSelectedCallback

The item selected callback.

GenericCallback * itemLockedInCallback

The item locked in callback.

GenericCallback * animationEndedCallback

The animation ended callback.

GenericCallback< int16_t > * itemPressedCallback

The item pressed callback.

· AnimationState currentAnimationState

The current animation state.

· int gestureStep

The gesture step.

· int gestureStepsTotal

The gesture steps total.

int gestureStart

The gesture start.

· int gestureEnd

The gesture end.

• int16_t xClick

The click.

int16_t yClick

The click.

• int32_t initialSwipeOffset

The initial swipe offset.

bool draggableX

Is the container draggable in the horizontal direction.

· bool draggableY

Is the container draggable in the vertical direction.

Additional Inherited Members

7.186.1 Detailed Description

A scroll base class with a list of drawables (DrawableList).

See also

ScrollWheelBase ScrollList DrawableList

7.186.2 Member Enumeration Documentation

7.186.2.1 AnimationState

```
enum AnimationState [protected]
```

Enumerator

NO_ANIMATION	No animation.
ANIMATING_GESTURE	Animating a gesture.
ANIMATING_DRAG	Animating a click+drag.

7.186.3 Constructor & Destructor Documentation

7.186.3.1 ScrollBase()

ScrollBase ()

Default constructor.

7.186.3.2 ∼ScrollBase()

```
~ScrollBase ( ) [inline], [virtual]
```

Destructor.

7.186.4 Member Function Documentation

7.186.4.1 allowHorizontalDrag()

Enables horizontal scrolling to be passed to the children. By default, scrolling in either direction is enabled. This function can be used to explicitly (dis)allow scrolling in the horizontal direction.

Parameters

(enable	If true (default), horizontal scrolling is enabled. If false, scrolling is disabled.	
---	--------	--	--

7.186.4.2 allowVerticalDrag()

```
void allowVerticalDrag (
          bool enable )
```

Enables the vertical scroll to be passed to the children. By default, scrolling in either direction is enabled. This function can be used to explicitly (dis)allow scrolling in the vertical direction.

Parameters

enable If true (default), vertical s	ing is enabled. If false, scrolling is disabled.
--------------------------------------	--

7.186.4.3 animateToltem()

Go to item, possibly with animation. The given item index is scrolled into view. If animationSteps is omitted, the default number of animation steps is used. If animationSteps is 0 no animation will be used, otherwise the number of animation steps specified is used.

Parameters

itemIndex	Zero-based index of the item.
animationSteps	(Optional) The steps to use for the animation. 0 means no animation. If omitted, default
	animation steps are used.

See also

setAnimationSteps

7.186.4.4 animateToPosition()

```
void animateToPosition ( int32\_t\ position, int16\_t\ steps = -1\ ) \ \ [protected], \ [virtual]
```

Animate to a new position/offset using the given number of steps.

Parameters

position	The new position.
steps	The steps.

Reimplemented in ScrollWheelBase.

7.186.4.5 getAnimationSteps()

```
uint16_t getAnimationSteps ( ) const
```

Gets animation steps as set in setAnimationSteps.

Returns

The animation steps.

See also

setAnimationSteps setEasingEquation

7.186.4.6 getCircular()

```
bool getCircular ( ) const [virtual]
```

Gets the circular setting, previously set using setCircular().

Returns

True if the list is circular (infinite), false if the list is not circular (finite).

See also

DrawableList::getCircular setCircular

```
7.186.4.7 getDragAcceleration()
uint16_t getDragAcceleration ( ) const
Gets drag acceleration (times 10).
Returns
     The drag acceleration.
Note
     The reason for multiplying the acceleration by 10 is to avoid introducing floating point arithmetics.
See also
     setDragAcceleration
7.186.4.8 getDrawableMargin()
int16_t getDrawableMargin ( ) const [virtual]
Gets drawable margin as set through the argument in setDrawables().
Returns
     The drawable margin.
7.186.4.9 getDrawableSize()
int16_t getDrawableSize ( ) const [virtual]
Gets drawable size as set through the first argument in setDrawables().
Returns
     The drawable size.
See also
     setDrawables
7.186.4.10 getHorizontal()
bool getHorizontal ( ) const [virtual]
Gets the orientation of the drawables, previously set using setHorizontal.
Returns
     True if it horizontal, false if it is vertical.
See also
```

DrawableList::getHorizontal

setHorizontal

7.186.4.11 getMaxSwipeItems()

```
uint16_t getMaxSwipeItems ( ) const
```

Gets maximum swipe items as set by setMaxSwipeItems.

Returns

The maximum swipe items, 0 means "no limit".

See also

setMaxSwipeItems

7.186.4.12 getNearestAlignedOffset()

Gets nearest offset aligned to a multiple of itemSize.

Parameters

```
offset The offset.
```

Returns

The nearest aligned offset.

Reimplemented in ScrollList.

7.186.4.13 getNormalizedOffset()

```
int getNormalizedOffset (
                int offset ) const [protected]
```

Gets normalized offset from a given offset from 0 down to -numltems*itemSize.

Parameters

```
offset The offset.
```

Returns

The normalized offset.

7.186.4.14 getNumberOfItems()

```
int16_t getNumberOfItems ( ) const [virtual]
```

Gets number of items in the DrawableList, as previously set using setNumberOfItems().

Returns

The number of items.

See also

setNumberOfItems DrawableList::getNumberOfItems

7.186.4.15 getOffset()

```
int32_t getOffset ( ) const [protected], [virtual]
```

Gets display offset of first item.

Returns

The offset.

7.186.4.16 getPositionForItem()

Get the position for an item. The position should ensure that the item is in view as defined by the semantics of the actual scroll class.

Parameters

itemIndex	Zero-based index of the item.
ILEITIITUEX	Leio-baseu iliuex oi lile ileili.

Returns

The position for item.

Implemented in ScrollList, and ScrollWheelBase.

7.186.4.17 getSwipeAcceleration()

```
uint16_t getSwipeAcceleration ( ) const
```

Gets swipe acceleration (times 10).

Returns

The swipe acceleration.

Note

The reason for multiplying the acceleration by 10 is to avoid introducing floating point arithmetics.

See also

setSwipeAcceleration

7.186.4.18 handleDragEvent()

Handles the drag event described by evt.

Parameters

```
evt The event.
```

Reimplemented from Drawable.

Reimplemented in ScrollWheelBase.

7.186.4.19 handleGestureEvent()

```
void handleGestureEvent ( {\tt const~GestureEvent~\&~evt~)} \quad [{\tt virtual}]
```

Handles the gesture event described by evt.

Parameters

```
evt The event.
```

Reimplemented from Drawable.

Reimplemented in ScrollWheelBase.

7.186.4.20 handleTickEvent()

```
void handleTickEvent ( ) [virtual]
```

Handles the tick event.

Reimplemented from Drawable.

7.186.4.21 initialize()

```
void initialize ( ) [inline], [virtual]
```

Initializes the contents of all drawables.

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.22 isAnimating()

```
bool isAnimating ( ) const
```

Query if this object is animating. This can be good to know if getSelectedItem() is called, as the result might not be as expected if isAnimating() returns true.

Returns

true if animating, false if not.

7.186.4.23 itemChanged()

Inform that an item has change and force all drawables with the given item index to be updated via the callback provided.

Parameters

	itemIndex	Zero-based index of the changed item.	
--	-----------	---------------------------------------	--

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.24 keepOffsetInsideLimits()

Parameters

newOffset	The new offset.
overShoot	The over shoot.

Returns

An int32_t.

Implemented in ScrollList, and ScrollWheelBase.

7.186.4.25 setAnimationEndedCallback()

Callback, called when the set animation ended.

Parameters

in,out	callback	The ended callback.

7.186.4.26 setAnimationSteps()

```
void setAnimationSteps ( int16\_t \ steps \ )
```

Sets animation steps when moving to a new selected item. The default value is 30.

Parameters

steps	The animation steps.
-1-1	

See also

setEasingEquation getAnimationSteps

7.186.4.27 setCircular()

```
void setCircular (
                bool circular ) [virtual]
```

Sets whether the list is circular (infinite) or not. A circular list is a list where the first drawable re-appears after the last item in the list - and the last item in the list appears before the first item in the list.

Parameters

circular	True if the list should be circular, false if the list should not be circular.
----------	--

See also

DrawableList::setCircular getCircular

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.28 setDragAcceleration()

Sets drag acceleration times 10, so "10" means "1", "15" means "1.5".

Parameters

acceleration	The drag acceleration. 10 makes the containers follow the finger, higher values makes the
	containers move faster.

Note

The reason for multiplying the acceleration by 10 is to avoid introducing floating point arithmetics.

See also

getDragAcceleration

7.186.4.29 setDrawableSize()

Sets drawables size. The drawable is is the size of each drawable in the list (this is enforced by the DrawableList class). The spacing is the amount of blank to add between each drawable. Half of the space is placed before and half of the space is placed after the drawable. The entire size of an item is thus size + spacing.

Parameters

drawableSize	The size of the drawable.	
drawableMargin	The margin around drawables (margin before and margin after).	

7.186.4.30 setEasingEquation()

Sets easing equation when changing the selected item, for example via swipe or AnimateTo.

Parameters

equation	The equation.

See also

setAnimationSteps getAnimationSteps

7.186.4.31 setHeight()

```
void setHeight (
                int16_t height ) [virtual]
```

Sets height of the ScrollBase.

Parameters

	height	The height.	The height is propagated to the child(ren).	•
--	--------	-------------	---	---

Reimplemented from Drawable.

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.32 setHorizontal()

Sets a horizontal layout. If horizontal is set true, all drawables are arranged side by side. If horizontal is set false, the drawables are arranged above and below each other (vertically).

Parameters

horizontal True to align drawables horizo	ntal, false to align drawables vertically.
---	--

Note

Default value is false, i.e. vertical layout.

See also

DrawableList::setHorizontal getHorizontal

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.33 setItemPressedCallback()

Set Callback which will be called when a item is pressed.

Parameters

in	callback	The callback.

7.186.4.34 setItemSelectedCallback()

Sets Callback which will be called when the selected item is clicked.

Parameters

```
in callback The callback.
```

7.186.4.35 setMaxSwipeItems()

Sets maximum swipe items. Often useful when there are five visible items on the screen and a swipe action should

at most swipe the next/previous five items into view to achieve sort of a paging effect.

Parameters

maxItems	The maximum items, 0 means "no limit".
----------	--

See also

getMaxSwipeItems

7.186.4.36 setNumberOfItems()

Sets number of items in the DrawableList. This forces all drawables to be updated to ensure that the content is correct. For example a minute selector might could have 60 items (only some of which are visible at any given time).

Parameters

items Number of items.

Note

The DrawableList is refreshed to reflect the change.

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.37 setOffset()

Sets display offset of first item.

Parameters

```
offset The offset.
```

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.38 setSwipeAcceleration()

Sets swipe acceleration (times 10).

Parameters

Note

The reason for multiplying the acceleration by 10 is to avoid introducing floating point arithmetics.

See also

getSwipeAcceleration

7.186.4.39 setWidth()

Sets width of the ScrollBase. The width is propagated to the child(ren).

Parameters

width	The width.

Reimplemented from Drawable.

Reimplemented in ScrollWheelWithSelectionStyle.

7.186.4.40 stopAnimation()

```
void stopAnimation ( )
```

Stops an animation if one is ongoing.

7.187 ScrollList Class Reference

A scrolling menu of drawables.

#include <touchgfx/containers/scrollers/ScrollList.hpp>

Public Member Functions

· ScrollList ()

Default constructor.

virtual ∼ScrollList ()

Destructor.

virtual void setDrawables (DrawableListItemsInterface &drawableListItems, GenericCallback
 ListItemsInterface *, int16_t, int16_t > &updateDrawableCallback)

Sets the drawables parameters.

void setWindowSize (int16 t items)

Sets window size.

void setPadding (int16_t paddingBefore, int16_t paddingAfter)

Sets distance offset before and after drawables in the ScrollList.

• int16 t getPaddingBefore () const

Gets distance before first drawable in ScrollList.

int16_t getPaddingAfter () const

Gets distance after last drawable in ScrollList.

void setSnapping (bool snap)

Sets snapping.

• bool getSnapping () const

Gets the current snap stetting.

• int16_t getItem (int16_t drawableIndex)

Gets an item.

virtual void handleClickEvent (const ClickEvent &evt)

Handles the click event described by evt.

Protected Member Functions

virtual int32_t getPositionForItem (int16_t itemIndex)

Gets position for an item.

virtual int32 t getNearestAlignedOffset (int32 t offset) const

Gets nearest aligned offset.

• virtual int32_t keepOffsetInsideLimits (int32_t newOffset, int16_t overShoot) const

Keep offset inside limits.

Protected Attributes

• int16_t paddingAfterLastItem

The distance after last item.

· bool snapping

True to snapping.

int windowSize

Size of the window.

Additional Inherited Members

7.187.1 Detailed Description

A scrolling menu of drawables. To preserve resources, a lot of items can be displayed using only a few drawables. To achieve this, please see DrawableList.

7.187.2 Constructor & Destructor Documentation

```
7.187.2.1 ScrollList()
ScrollList ( )
Default constructor.

7.187.2.2 ~ScrollList()
~ScrollList ( ) [inline], [virtual]
Destructor.
```

7.187.3 Member Function Documentation

7.187.3.1 getItem()

Gets an item.

Parameters

drawableIndex Zero-based index of the drawable
--

Returns

The item.

7.187.3.2 getNearestAlignedOffset()

Parameters

offset The offset.

Returns

The nearest aligned offset.

Reimplemented from ScrollBase.

7.187.3.3 getPaddingAfter()

```
int16_t getPaddingAfter ( ) const
```

Gets distance after last drawable in ScrollList.

Returns

The distance after the last drawable in the ScrollList.

See also

```
setPadding
getPaddingBefore
```

7.187.3.4 getPaddingBefore()

```
int16_t getPaddingBefore ( ) const
```

Gets distance before first drawable in ScrollList.

Returns

The distance before.

See also

```
setPadding
getPaddingAfter
```

7.187.3.5 getPositionForItem()

Get the position for an item. The position should ensure that the item is in view as defined by the semantics of the actual scroll class. If the item is already in view, the current offset is returned and not the offset of the given item.

Parameters

	itemIndex	Zero-based index of the item.	
--	-----------	-------------------------------	--

Returns

The position for item.

Implements ScrollBase.

7.187.3.6 getSnapping()

```
bool getSnapping ( ) const
```

Gets the current snap stetting.

Returns

true if snapping is set, false otherwise.

7.187.3.7 handleClickEvent()

Parameters

evt	The event.
evi	THE EVEIL.

Reimplemented from Drawable.

7.187.3.8 keepOffsetInsideLimits()

Parameters

newOffset	The new offset.
overShoot	The over shoot.

Returns

An int32_t.

Implements ScrollBase.

7.187.3.9 setDrawables()

Sets the drawables parameters. These parameters are

- · The access class to the array of drawables
- · Callback to update the contents of a drawable.

Parameters

in	,out	drawableListItems	Number of drawables allocated.
in		updateDrawableCallback	A callback to update the contents of a drawable.

See also

DrawableList::setDrawables

7.187.3.10 setPadding()

Sets distance offset before and after the "visible" drawables in the ScrollList. This allows the actual area where widgets are placed to have a little extra area where parts of drawables can be seen. For example if the ScrollList is 200, each drawable is 50 and distance before and distance after are 25, then there is room for three visible

drawables inside the ScrollList. When scrolling, part of the scrolled out drawables can be seen before and after the three drawables. Actually 25/50 = 50% of a drawable can be seen before and after the three drawables in the ScrollList.

Parameters

distanceBefore	The distance before the first drawable in the ScrollList.
distanceAfter	The distance after the last drawable in the ScrollList.

See also

```
getPaddingBefore
getPaddingAfter
```

7.187.3.11 setSnapping()

```
void setSnapping (
          bool snap )
```

Set snapping. If snapping is false, the items can flow freely. If snapping is true, the items will snap into place so an item is always in the "selected" spot.

Parameters

snap	true to snap.
------	---------------

7.187.3.12 setWindowSize()

Sets window size. This is the number of items that should always be visible. The default value is 1.

Parameters

items	The items.

Note

This only applies to non-circular lists.

7.188 ScrollWheel Class Reference

A scroll wheel.

```
#include <touchgfx/containers/scrollers/ScrollWheel.hpp>
```

Public Member Functions

• ScrollWheel ()

Default constructor.

virtual ∼ScrollWheel ()

Destructor.

virtual void setDrawables (DrawableListItemsInterface &drawableListItems, GenericCallback
 Drawable←
 ListItemsInterface *, int16_t, int16_t > &updateDrawableCallback)

Sets the drawables.

Additional Inherited Members

7.188.1 Detailed Description

A scroll wheel is a list of identically styled drawables which can be scrolled through. One of the items in the list is the "selected" one, and scrolling through the list can be done in various ways. The ScrollWheel uses the DrawableList to make it possible to handle a huge number of items using only a limited number of drawables by reusing drawables that are no longer in view.

See also

DrawableList ScrollWheelWithSelectionStyle

7.188.2 Constructor & Destructor Documentation

```
7.188.2.1 ScrollWheel()
```

```
ScrollWheel ()
```

Default constructor.

```
7.188.2.2 \simScrollWheel()
```

```
\simScrollWheel ( ) [virtual]
```

Destructor.

7.188.3 Member Function Documentation

7.188.3.1 setDrawables()

Sets the drawables used by the scroll wheel. The drawables are accessed through a callback that will return the needed drawable and another callback that will put the right data in the drawable.

Parameters

in,out	drawableListItems	Number of drawables.
in,out	updateDrawableCallback	The update drawable callback.

7.189 ScrollWheelBase Class Reference

A scroll wheel base class.

#include <touchgfx/containers/scrollers/ScrollWheelBase.hpp>

Public Member Functions

ScrollWheelBase ()

Default constructor.

virtual ∼ScrollWheelBase ()

Destructor.

virtual void setSelectedItemOffset (int16 t offset)

Sets selected item offset.

virtual int16_t getSelectedItemOffset () const

Gets selected item offset.

• int getSelectedItem () const

Gets selected item.

• virtual int32_t keepOffsetInsideLimits (int32_t newOffset, int16_t overShoot) const

Keep offset inside limits.

virtual void handleClickEvent (const ClickEvent &evt)

Handles the click event described by evt.

virtual void handleDragEvent (const DragEvent &evt)

Handles the drag event described by evt.

• virtual void handleGestureEvent (const GestureEvent &evt)

Handles the gesture event described by evt.

void setAnimateToCallback (GenericCallback< int16_t > &callback)

Sets Callback which will be called when the ScrollWheel animates to a new item.

Protected Member Functions

virtual int32_t getPositionForItem (int16_t itemIndex)

Gets position for an item.

virtual void animateToPosition (int32_t position, int16_t steps=-1)

Animate to a new position/offset using the given number of steps.

Protected Attributes

GenericCallback
 int16_t > * animateToCallback

The animate to callback.

Additional Inherited Members

7.189.1 Detailed Description

A scroll wheel base class. Used by ScrollWheel and ScrollWheelWithHighlight.

See also

ScrollWheel

ScrollWheelWithHighlight

7.189.2 Constructor & Destructor Documentation

7.189.2.1 ScrollWheelBase()

```
ScrollWheelBase ( )
```

Default constructor.

7.189.2.2 ∼ScrollWheelBase()

```
~ScrollWheelBase ( ) [inline], [virtual]
```

Destructor.

7.189.3 Member Function Documentation

7.189.3.1 animateToPosition()

```
void animateToPosition ( int32\_t\ position, int16\_t\ steps = -1\ ) \ \ [protected], \ [virtual]
```

Animate to a new position/offset using the given number of steps.

Parameters

position	The new position.	
steps	(Optional) The number of steps. If steps is $<$ 0, the default number of steps is used (this is the	
	default). If steps ==0, there will be no animation, simply a direct skip to the given position.	

Reimplemented from ScrollBase.

7.189.3.2 getPositionForItem()

Get the position for an item. The position should ensure that the item is in view as defined by the semantics of the actual scroll class.

Parameters

itamInday	Zero-based index of the item.
Hellinaex	Lero-based index of the item.
ILCITITIOCX	2010 based index of the item.

Returns

The position for item.

Implements ScrollBase.

7.189.3.3 getSelectedItem()

```
int getSelectedItem ( ) const
```

Gets selected item. If an animation is in progress, the item that is being scrolled to is returned, not the item that happens to be flying by at the time.

Returns

The selected item.

7.189.3.4 getSelectedItemOffset()

```
int16_t getSelectedItemOffset ( ) const [virtual]
```

Gets offset of selected item measured in pixels relative to the start of the widget.

Returns

The selected item offset.

See also

setSelectedItemOffset

7.189.3.5 handleClickEvent()

Handles the click event described by evt.

Parameters

```
evt The event.
```

Reimplemented from Drawable.

7.189.3.6 handleDragEvent()

Handles the drag event described by evt.

Parameters

evt The event.

Reimplemented from ScrollBase.

7.189.3.7 handleGestureEvent()

Handles the gesture event described by evt.

Parameters

```
evt The event.
```

Reimplemented from ScrollBase.

7.189.3.8 keepOffsetInsideLimits()

Parameters

newOffset	The new offset.
overShoot	The over shoot.

Returns

An int32_t.

Implements ScrollBase.

7.189.3.9 setAnimateToCallback()

Sets Callback which will be called when the ScrollWheel animates to a new item.

Parameters

in	callback	The callback.
----	----------	---------------

7.189.3.10 setSelectedItemOffset()

```
void setSelectedItemOffset (
                int16_t offset ) [virtual]
```

Sets selected item offset. This is the number of pixels from the start of the widget where the selected item is placed on screen. The offset is the relative x coordinate if the ScrollWheel is horizontal, otherwise it is the relative y coordinate. If this value is zero, the selected item is placed at the very start of the widget.

Reimplemented in ScrollWheelWithSelectionStyle.

7.190 ScrollWheelWithSelectionStyle Class Reference

A scroll wheel with selection style.

#include <touchgfx/containers/scrollers/ScrollWheelWithSelectionStyle.hpp>

Public Member Functions

• ScrollWheelWithSelectionStyle ()

Default constructor.

• virtual ~ScrollWheelWithSelectionStyle ()

Destructor.

virtual void setWidth (int16_t width)

Sets the width.

virtual void setHeight (int16_t height)

Sets the height.

virtual void setHorizontal (bool horizontal)

Sets whether the scroll wheel is horizontal or vertical.

virtual void setCircular (bool circular)

Sets whether the scroll wheel is circular.

virtual void setNumberOfItems (int16_t numberOfItems)

Sets number of items in the scroll wheel.

virtual void setSelectedItemOffset (int16_t offset)

Sets selected item offset.

virtual void setSelectedItemExtraSize (int16_t extraSizeBefore, int16_t extraSizeAfter)

Sets selected item extra size.

virtual int16_t getSelectedItemExtraSizeBefore () const

Gets selected item extra size before.

virtual int16_t getSelectedItemExtraSizeAfter () const

Gets selected item extra size after.

virtual void setSelectedItemMargin (int16_t marginBefore, int16_t marginAfter)

Sets margin around selected item.

· virtual int16 t getSelectedItemMarginBefore () const

Gets selected item margin before.

virtual int16_t getSelectedItemMarginAfter () const

Gets selected item margin after.

virtual void setSelectedItemPosition (int16_t offset, int16_t extraSizeBefore, int16_t extraSizeAfter, int16_t marginBefore, int16_t marginAfter)

Sets the selected item offset.

virtual void setDrawableSize (int16_t drawableSize, int16_t drawableMargin)

Sets drawable size.

virtual void setDrawables (DrawableListItemsInterface &drawableListItems, GenericCallback
 ListItemsInterface *, int16_t, int16_t > &updateDrawableCallback, DrawableListItemsInterface ¢er←
 DrawableListItems, GenericCallback
 DrawableListItemsInterface *, int16_t, int16_t > &updateCenter←
 DrawableCallback)

Setups the widget.

· virtual void itemChanged (int itemIndex)

Item changed.

virtual void initialize ()

Initializes the contents of all drawables.

Protected Member Functions

· virtual void setOffset (int32_t offset)

Sets offset of item 0 relative to the selected item's position.

• void refreshDrawableListsLayout ()

Refresh drawable lists layout.

Protected Attributes

· int16_t drawablesInFirstList

List of drawables in firsts.

DrawableList list1

The center list.

DrawableList list2

The last list.

• int16 t extraSizeBeforeSelectedItem

The distance before selected item.

• int16_t extraSizeAfterSelectedItem

The distance after selected item.

int16 t marginBeforeSelectedItem

The distance before selected item.

int16_t marginAfterSelectedItem

The distance after selected item.

• DrawableListItemsInterface * drawables

The drawables at the beginning and end of the scroll wheel.

• DrawableListItemsInterface * centerDrawables

The drawables at the center of the scroll wheel.

 $\bullet \quad \text{GenericCallback} < \text{DrawableListItemsInterface} *, \text{int16_t}, \text{int16_t} > * \text{originalUpdateDrawableCallback}$

The original update drawable callback.

GenericCallback
 DrawableListItemsInterface *, int16_t, int16_t > * originalUpdateCenterDrawable←
 Callback

The original update center drawable callback.

Additional Inherited Members

7.190.1 Detailed Description

A scroll wheel with selection style. Similar to an ordinary ScrollWheel, but with a different style for the selected item which can thus be bold, have a different color or similar effect to highlight it.

See also

DrawableList ScrollWheel

7.190.2 Constructor & Destructor Documentation

```
7.190.2.1 ScrollWheelWithSelectionStyle()
```

```
ScrollWheelWithSelectionStyle ( )
```

Default constructor.

7.190.2.2 ~ScrollWheelWithSelectionStyle()

```
\simScrollWheelWithSelectionStyle ( ) [inline], [virtual]
```

Destructor.

7.190.3 Member Function Documentation

7.190.3.1 getSelectedItemExtraSizeAfter()

```
int16_t getSelectedItemExtraSizeAfter ( ) const [virtual]
```

Gets selected item extra size after.

Returns

The selected item extra size after.

See also

setSelectedItemExtraSize

7.190.3.2 getSelectedItemExtraSizeBefore()

```
int16_t getSelectedItemExtraSizeBefore ( ) const [virtual]
```

Gets selected item extra size before.

Returns

The selected item extra size before.

See also

setSelectedItemExtraSize

7.190.3.3 getSelectedItemMarginAfter()

```
int16_t getSelectedItemMarginAfter ( ) const [virtual]
```

Gets selected item margin after.

Returns

The selected item margin after.

See also

setSelectedItemMargin

7.190.3.4 getSelectedItemMarginBefore()

```
int16_t getSelectedItemMarginBefore ( ) const [virtual]
```

Gets selected item margin before.

Returns

The selected item margin before.

See also

setSelectedItemMargin

7.190.3.5 initialize()

```
void initialize ( ) [inline], [virtual]
```

Initializes the contents of all drawables.

Reimplemented from ScrollBase.

7.190.3.6 itemChanged()

Inform that an item has change and force all drawables with the given item index to be updated via the callback provided.

Parameters

itemIndex Zero-based index of the changed item.

Reimplemented from ScrollBase.

7.190.3.7 refreshDrawableListsLayout()

```
void refreshDrawableListsLayout ( ) [protected]
```

Refresh drawable lists layout. Ensure that the three DrawableLists are places correctly and setup properly. This is typically done after the ScrollWheelWithSelectionStyle has been resized or the size of the selected item is changed.

7.190.3.8 setCircular()

Sets whether the scroll wheel is circular. IF the scroll wheel is circular, it can be scrolled infinitely so that the last item appears before the first item, just like the first item appears after the last item in the list.

Parameters

ircular True to make the scroll wheel circular.

See also

getCircular

Reimplemented from ScrollBase.

7.190.3.9 setDrawables()

Setups the widget. Numerous parameters control the position of the widget, the two scroll lists inside and the values in them.

Parameters

in,out	drawableListItems	Number of drawables in outer array.
in	updateDrawableCallback	The callback to update a drawable.
in,out	centerDrawableListItems	Number of drawables in center array.
in	updateCenterDrawableCallback	The callback to update a center drawable.

7.190.3.10 setDrawableSize()

Sets drawable size. Each item in the scroll wheel will have a size of the sum of the two numbers, where the drawableSize is the size of the drawable in the list and the drawableMargin is the margin between each drawable (half of which is placed before the drawable, the rest is placed after the drawable).

drawableSize	Size of the drawable.
drawableMargin	The drawable margin.

See also

```
getDrawableSize
getDrawableMargin
getItemSize
```

7.190.3.11 setHeight()

```
void setHeight (
          int16_t height ) [virtual]
```

Sets the height. If the scroll wheel is horizontal, the height is propagated to all the drawables.

Parameters

height The height.

Reimplemented from ScrollBase.

7.190.3.12 setHorizontal()

```
void setHorizontal (
                bool horizontal ) [virtual]
```

Sets whether the scroll wheel is horizontal or vertical. If the scroll wheel is hortizontal, the items are arranged side by side, otherwise they are arranged above and below each other.

Parameters

horizonta	<i>1</i> 7	True means	horizontal,	false	means	vertical.
-----------	------------	------------	-------------	-------	-------	-----------

See also

getHorizontal

Reimplemented from ScrollBase.

7.190.3.13 setNumberOfItems()

Sets number of items in the scroll wheel. The scroll wheel is refreshed to ensure that everything is displayed properly on the screen.

numberOfItems Number of items

See also

getNumberOfItems

Reimplemented from ScrollBase.

7.190.3.14 setOffset()

Sets offset of item 0 relative to the selected item's position.

Parameters

offset	The offset.
011001	1110 011001.

See also

getOffset

Reimplemented from ScrollBase.

7.190.3.15 setSelectedItemExtraSize()

Sets selected item extra size to make the size of the area for the center drawables larger.

Parameters

extraSizeBefore	The extra size before.
extraSizeAfter	The extra size after.

See also

setSelectedItemOffset

7.190.3.16 setSelectedItemMargin()

Sets margin around selected item. This like an invisible area added before and after the selected item (including extra size).

marginBefore	The margin before.
marginAfter	The margin after.

See also

setSelectedItemOffset setSelectedItemExtraSize

7.190.3.17 setSelectedItemOffset()

```
void setSelectedItemOffset (
                int16_t offset ) [virtual]
```

Sets selected item offset. This is the number of pixels from the start of the widget where the selected item is placed on screen. The offset is the relative x coordinate if the ScrollWheel is horizontal, otherwise it is the relative y coordinate. If this value is zero, the selected item is placed at the very start of the widget.

Parameters

offset	The offset.
--------	-------------

Reimplemented from ScrollWheelBase.

7.190.3.18 setSelectedItemPosition()

Sets the selected item offset. This is the distance from the beginning of the ScrollWheel measured in pixels. The distance before and after that should also be drawn using the center drawables - for example to extend area of emphasized elements - can also be specified. Further, if a gap is needed between the "normal" drawables and the center drawables - for example to give the illusion that items disappear under a graphical element, only to appear in the center.

This is a combination of setSelectedItemOffset, setSelectedItemExtraSize and setSelectedItemMargin.

Parameters

offset	The offset of the selected item.
extraSizeBefore	The extra size before the selected item.
extraSizeAfter	The extra size after the selected item.
marginBefore	The margin before the selected item.
marginAfter	The margin after the selected item.

See also

setSelectedItemOffset setSelectedItemExtraSize setSelectedItemMargin

7.190.3.19 setWidth()

```
void setWidth (
          int16_t width ) [virtual]
```

Sets the width. If the scroll wheel is vertical, the width is propagated to all the drawables.

Parameters

```
width The width.
```

Reimplemented from ScrollBase.

7.191 SDL2TouchController Class Reference

TouchController for the simulator.

```
#include <platform/driver/touch/SDL2TouchController.hpp>
```

Public Member Functions

· virtual void init ()

Initializes touch controller.

virtual bool sampleTouch (int32_t &x, int32_t &y)

Checks whether the touch screen is being touched, and if so, what coordinates.

7.191.1 Detailed Description

TouchController for the simulator.

See also

TouchController

7.191.2 Member Function Documentation

```
7.191.2.1 init()
void init ( ) [virtual]
```

Initializes touch controller.

Implements TouchController.

7.191.2.2 sampleTouch()

Checks whether the touch screen is being touched, and if so, what coordinates.

Parameters

out	Х	The x position of the touch
out	у	The y position of the touch

Returns

True if a touch has been detected, otherwise false.

Implements TouchController.

7.192 SDLTouchController Class Reference

TouchController for the simulator.

```
#include <platform/driver/touch/SDLTouchController.hpp>
```

Public Member Functions

· virtual void init ()

Initializes touch controller.

virtual bool sampleTouch (int32_t &x, int32_t &y)

Checks whether the touch screen is being touched, and if so, what coordinates.

7.192.1 Detailed Description

TouchController for the simulator.

See also

TouchController

7.192.2 Member Function Documentation

```
7.192.2.1 init()
void init ( ) [virtual]
```

Initializes touch controller.

Implements TouchController.

7.192.2.2 sampleTouch()

Checks whether the touch screen is being touched, and if so, what coordinates.

Parameters

out	X	The x position of the touch
out	у	The y position of the touch

Returns

True if a touch has been detected, otherwise false.

Implements TouchController.

7.193 Shape < POINTS > Class Template Reference

Simple widget capable of drawing a shape.

#include <touchgfx/widgets/canvas/Shape.hpp>

Public Member Functions

virtual ∼Shape ()

Virtual Destructor.

• virtual int getNumPoints () const

Gets number points used to make up the shape.

virtual void setCorner (int i, CWRUtil::Q5 x, CWRUtil::Q5 y)

Sets a corner of the shape in Q5 format.

virtual CWRUtil::Q5 getCornerX (int i) const

Gets the x coordinate of a corner.

virtual CWRUtil::Q5 getCornerY (int i) const

Gets the y coordinate of a corner.

Protected Member Functions

virtual void setCache (int i, CWRUtil::Q5 x, CWRUtil::Q5 y)

Sets the cached coordinates of a given corner.

virtual CWRUtil::Q5 getCacheX (int i) const

Gets cached x coordinate of a corner.

· virtual CWRUtil::Q5 getCacheY (int i) const

Gets cached y coordinate of a corner.

Additional Inherited Members

7.193.1 Detailed Description

```
template<uint16_t POINTS>
class touchgfx::Shape< POINTS >
```

Simple widget capable of drawing a shape. The shape can be scaled and rotated around 0,0. Note that the y axis goes down, so a shape that goes up must be given negative coordinates.

The Shape class allows the user to draw any shape and allows the defined shape to be scaled, rotated and moved freely. Example uses could be the hands of a clock (see the touchgfx_demo2014 for an actual implementation).

Template Parameters

POINTS	The number of points in the given shape.
T	The type of the points used for the shape. Must be int or float.

See also

CanvasWidget

7.193.2 Constructor & Destructor Documentation

7.193.3 Member Function Documentation

7.193.3.1 getCacheX()

```
CWRUtil::Q5 getCacheX (
                int i ) const [inline], [protected], [virtual]
```

Gets cached x coordinate of a corner.

Parameters

```
\overline{i} Zero-based index of the corner.
```

Returns

The cached x coordinate.

Implements AbstractShape.

7.193.3.2 getCacheY()

```
CWRUtil::Q5 getCacheY (
                int i ) const [inline], [protected], [virtual]
```

Gets cached y coordinate of a corner.

Parameters

i Zero-based index of the corner.

Returns

The cached y coordinate.

Implements AbstractShape.

```
7.193.3.3 getCornerX()
```

```
CWRUtil::Q5 getCornerX (
                int i ) const [inline], [virtual]
```

Gets the x coordinate of a corner.

Parameters

i Zero-based index of the corner.

Returns

The corner x coordinate.

Implements AbstractShape.

7.193.3.4 getCornerY()

```
CWRUtil::Q5 getCornerY (
                int i ) const [inline], [virtual]
```

Gets the y coordinate of a corner.

Parameters

```
i Zero-based index of the corner.
```

Returns

The corner y coordinate.

Implements AbstractShape.

```
7.193.3.5 getNumPoints()
```

```
int getNumPoints ( ) const [inline], [virtual]
```

Gets number points used to make up the shape.

Returns

The number points.

Implements AbstractShape.

7.193.3.6 setCache()

Sets the cached coordinates of a given corner. The coordinates in the cache are the coordinates from the corners after rotating and scaling the coordinate.

Parameters

i	Zero-based index of the corner.	
Х	The x coordinate.	
У	The y coordinate.	

Implements AbstractShape.

7.193.3.7 setCorner()

Sets a corner of the shape in Q5 format.

Parameters

i	Zero-based index of the corner.	
Χ	The x coordinate in Q5 format.	
У	The y coordinate in Q5 format.	

Implements AbstractShape.

7.194 AbstractShape::ShapePoint < T > Struct Template Reference

Defines an alias representing the array of points making up the abstract shape.

```
#include <touchgfx/widgets/canvas/AbstractShape.hpp>
```

Public Attributes

T x

The x coordinate of the points.

• T y

The y coordinate of the points.

7.194.1 Detailed Description

```
\label{template} \mbox{template} < \mbox{typename T} > \\ \mbox{struct touchgfx::AbstractShape::ShapePoint} < \mbox{T} > \\ \mbox{}
```

Defines an alias representing the array of points making up the abstract shape. This will help setting up the abstractShape very easily using setAbstractShape().

Template Parameters

```
T Generic type parameter, either int or float.
```

See also

setAbstractShape()

7.195 SingleBlockAllocator < block_size, bytes_pr_pixel > Class Template Reference

This class is partial framebuffer allocator using just one block.

```
#include <touchgfx/hal/FrameBufferAllocator.hpp>
```

Public Member Functions

 virtual uint16_t allocateBlock (const uint16_t x, const uint16_t y, const uint16_t width, const uint16_t height, uint8_t **block)

Allocates a framebuffer block.

virtual void markBlockReadyForTransfer ()

Marks a previously allocated block as ready to be transferred to the LCD.

• virtual bool hasBlockReadyForTransfer ()

Check if a block is ready for transfer to the LCD.

virtual const uint8_t * getBlockForTransfer (Rect &rect)

Get the block ready for transfer.

• virtual void freeBlockAfterTransfer ()

Free a block after transfer to the LCD.

7.195.1 Detailed Description

```
template<uint16_t block_size, uint32_t bytes_pr_pixel> class touchgfx::SingleBlockAllocator< block_size, bytes_pr_pixel>
```

This class is partial framebuffer allocator using just one block. No new buffer can be allocated until the block has been transferred to LCD.

See also

ManyBlockAllocator

7.195.2 Member Function Documentation

7.195.2.1 allocateBlock()

Allocates a framebuffer block. The block will have at least the width requested. The height of the allocated block can be lower than requested if not enough memory is available. This class calls FrameBufferAllocatorWaitOnTransfer() if no block is available.

Parameters

X	The absolute x coordinate of the block on the screen.	
У	The absolute y coordinate of the block on the screen.	
width	The width of the block.	
height	The height of the block.	
block	Pointer to pointer to return the block address in.	

Returns

The height of the allocated block.

Implements FrameBufferAllocator.

7.195.2.2 freeBlockAfterTransfer()

```
void freeBlockAfterTransfer ( ) [inline], [virtual]
```

Marks a previously allocated block as transferred and ready to reuse.

Implements FrameBufferAllocator.

7.195.2.3 getBlockForTransfer()

Get the block ready for transfer.

Parameters

rect	Reference to rect to write block x, y, width, and height.
------	---

Returns

Returns the address of the block ready for transfer.

Implements FrameBufferAllocator.

7.195.2.4 hasBlockReadyForTransfer()

```
bool hasBlockReadyForTransfer ( ) [inline], [virtual]
```

Check if a block is ready for transfer to the LCD.

Returns

True if a block is ready for transfer.

Implements FrameBufferAllocator.

7.195.2.5 markBlockReadyForTransfer()

```
void markBlockReadyForTransfer ( ) [inline], [virtual]
```

Marks a previously allocated block as ready to be transferred to the LCD.

Implements FrameBufferAllocator.

7.196 SlideMenu Class Reference

SlideMenu is a container that has the functionality of being either collapsed or expanded.

```
#include <touchgfx/containers/SlideMenu.hpp>
```

Public Types

• enum State { COLLAPSED, EXPANDED }

Values that represent the SlideMenu states.

enum ExpandDirection { SOUTH, NORTH, EAST, WEST }

Values that represent the expand directions.

Public Member Functions

· SlideMenu ()

Default constructor.

virtual ∼SlideMenu ()

Destructor.

• virtual void setup (SlideMenu::ExpandDirection newExpandDirection, const Bitmap &backgroundBMP, const Bitmap &stateChangeButtonBMP, const Bitmap &stateChangeButtonPressedBMP)

Setup the SlideMenu by positioning the stateChangeButton next to background image relative to the expand direction and center it in the other dimension.

virtual void setup (SlideMenu::ExpandDirection newExpandDirection, const Bitmap &backgroundBMP, const Bitmap &stateChangeButtonPressedBMP, int16_t backgroundX, int16 t backgroundY, int16 t stateChangeButtonX, int16 t stateChangeButtonY)

Setup method for the SlideMenu. Positioning of the background image and the stateChangeButton is done by stating the X and Y coordinates for the elements (relative to the SlideMenu).

virtual void setExpandDirection (SlideMenu::ExpandDirection newExpandDirection)

Sets the expand direction.

virtual SlideMenu::ExpandDirection getExpandDirection () const

Gets the expand direction.

virtual void setVisiblePixelsWhenCollapsed (int16_t visiblePixels)

Sets the amount of visible pixels when collapsed.

virtual int16_t getVisiblePixelsWhenCollapsed () const

Gets the visible pixels when collapsed.

virtual void setHiddenPixelsWhenExpanded (int16_t hiddenPixels)

Sets the amount of hidden pixels when expanded.

• virtual int16_t getHiddenPixelsWhenExpanded () const

Gets the hidden pixels when expanded.

virtual void setExpandedStateTimeout (uint16_t timeout)

Sets the expanded state timeout in ticks.

virtual uint16_t getExpandedStateTimeout () const

Gets expanded state timeout.

virtual void setAnimationDuration (uint16 t duration)

Sets the animation duration.

• virtual uint16 t getAnimationDuration () const

Gets the animation duration.

virtual void setAnimationEasingEquation (EasingEquation animationEasingEquation)

Sets the animation easing equation.

virtual EasingEquation getAnimationEasingEquation () const

Gets the animation easing equation.

• virtual void setState (SlideMenu::State newState)

Sets the state of the SlideMenu. No animation is performed.

virtual void animateToState (SlideMenu::State newState)

Animate to the given state.

virtual SlideMenu::State getState ()

Gets the current state.

virtual void resetExpandedStateTimer ()

Resets the expanded state timer.

virtual uint16_t getExpandedStateTimer () const

Gets the expanded state timer.

virtual int16_t getBackgroundX () const

Gets the background x coordinate.

virtual int16_t getBackgroundY () const

Gets the background y coordinate.

virtual int16_t getStateChangeButtonX () const

Gets the state change button x coordinate.

virtual int16_t getStateChangeButtonY () const

Gets the state change button y coordinate.

virtual void setStateChangedCallback (GenericCallback < const SlideMenu & > &callback)

Set the state changed callback. This callback is called when the state change button is clicked.

virtual void setStateChangedAnimationEndedCallback (GenericCallback
 const SlideMenu & > &callback)

Set the state change animation ended callback. This callback is called when a state change animation has ended.

virtual void add (Drawable &d)

Adds a drawable to the container.

virtual void remove (Drawable &d)

Removes the drawable from the container.

Protected Member Functions

• void stateChangeButtonClickedHandler (const AbstractButton &button)

Handler for the state change button clicked event.

void animationEndedHandler (const MoveAnimator< Container > &container)

Handler for the state change animation ended event.

· virtual void handleTickEvent ()

Handles the tick event.

virtual int16 t getCollapsedXCoordinate ()

Gets the x coordinate for the collapsed state.

virtual int16_t getCollapsedYCoordinate ()

Gets the y coordinate for the collapsed state.

virtual int16_t getExpandedXCoordinate ()

Gets the x coordinate for the expanded state.

virtual int16_t getExpandedYCoordinate ()

Gets the y coordinate for the expanded state.

Protected Attributes

MoveAnimator < Container > menuContainer

The container holding the actual menu items. This is the container that performs the state change animation.

· Button stateChangeButton

The state change button that toggles the SlideMenu state.

· Image background

The background of the SlideMenu.

• Callback< SlideMenu, const AbstractButton & > onStateChangeButtonClicked

The local state changed button clicked callback.

Callback
 SlideMenu, const MoveAnimator
 Container > & > animationEndedCallback

The local state changed animation ended callback.

GenericCallback< const SlideMenu &> * stateChangedCallback

The public state changed button clicked callback.

• GenericCallback< const SlideMenu &> * stateChangedAnimationEndedCallback

The public state changed animation ended callback.

· SlideMenu::State currentState

The current state of the SlideMenu.

· SlideMenu::ExpandDirection expandDirection

The expand direction of the SlideMenu.

EasingEquation animationEquation

The easing equation used for the state change animation.

int16_t visiblePixelsWhenCollapsed

The number of visible pixels when collapsed.

int16_t hiddenPixelsWhenExpanded

The number of hidden pixels when expanded.

· uint16_t expandedStateTimeout

The expanded state timeout.

uint16 t expandedStateTimer

The timer that counts towards the expandedStateTimeout. If reached the SlideMenu will animate to COLLAPSED.

• uint16_t animationDuration

The animation duration of the state change animation.

Additional Inherited Members

7.196.1 Detailed Description

SlideMenu is a container that has the functionality of being either collapsed or expanded. The SlideMenu consists of a background and a activate button that toggles the SlideMenus collapsed/expanded state.

The relative positions of the background and state change button is configurable as is the direction in which the SlideMenu expands and collapses. How much of the SlideMenu that is visible when collapsed can be set with the setVisiblePixelsWhenCollapsed(..) method. It is, of course, important that the state change button is accessible when collapsed. The SlideMenu will animate back to the collapsed state after a expandedStateTimeout is reached. The timer can be reset, for example when the user interacts with elements in the list. Use the resetExpanded StateTimer(..) method for this.

Actual menu elements are added normally using the add(...) method and are positioned relative to the SlideMenu.

7.196.2 Member Function Documentation

Reimplemented from Container.

7.196.2.2 animateToState()

Parameters

newState	The new state of the SlideMenu.
----------	---------------------------------

7.196.2.3 animationEndedHandler()

```
void animationEndedHandler ( const\ MoveAnimator <\ Container\ >\ \&\ container\ ) \quad [protected]
```

Parameters

container The menuContainer.

```
7.196.2.4 getAnimationDuration()
```

```
uint16_t getAnimationDuration ( ) const [virtual]
```

Returns

The animation duration.

7.196.2.5 getAnimationEasingEquation()

```
EasingEquation getAnimationEasingEquation ( ) const [virtual]
```

Returns

The animation easing equation.

7.196.2.6 getBackgroundX()

```
int16_t getBackgroundX ( ) const [virtual]
```

Returns

The background x coordinate.

7.196.2.7 getBackgroundY()

```
int16_t getBackgroundY ( ) const [virtual]
```

Returns

The background y coordinate.

7.196.2.8 getCollapsedXCoordinate()

```
int16_t getCollapsedXCoordinate ( ) [protected], [virtual]
```

Returns

The collapsed x coordinate.

7.196.2.9 getCollapsedYCoordinate()

```
int16_t getCollapsedYCoordinate ( ) [protected], [virtual]
```

Returns

The collapsed y coordinate.

7.196.2.10 getExpandDirection()

```
SlideMenu::ExpandDirection getExpandDirection ( ) const [virtual]
```

Returns

The expand direction.

7.196.2.11 getExpandedStateTimeout()

```
uint16_t getExpandedStateTimeout ( ) const [virtual]
```

Returns

The expanded state timeout.

7.196.2.12 getExpandedStateTimer()

```
uint16_t getExpandedStateTimer ( ) const [virtual]
```

Returns

The expanded state timer.

7.196.2.13 getExpandedXCoordinate()

```
int16_t getExpandedXCoordinate ( ) [protected], [virtual]
```

Returns

The expanded x coordinate.

7.196.2.14 getExpandedYCoordinate()

```
int16_t getExpandedYCoordinate ( ) [protected], [virtual]
```

Returns

The expanded y coordinate.

7.196.2.15 getHiddenPixelsWhenExpanded()

```
int16_t getHiddenPixelsWhenExpanded ( ) const [virtual]
```

Returns

The hidden pixels when expanded.

7.196.2.16 getState()

```
SlideMenu::State getState ( ) [virtual]
```

Returns

The current state.

7.196.2.17 getStateChangeButtonX()

```
int16_t getStateChangeButtonX ( ) const [virtual]
```

Returns

The state change button x coordinate.

7.196.2.18 getStateChangeButtonY()

```
int16_t getStateChangeButtonY ( ) const [virtual]
```

Returns

The state change button y coordinate.

7.196.2.19 getVisiblePixelsWhenCollapsed()

```
int16_t getVisiblePixelsWhenCollapsed ( ) const [virtual]
```

Returns

The visible pixels when collapsed.

7.196.2.20 remove()

Parameters

d The drawable to remove.

Reimplemented from Container.

7.196.2.21 resetExpandedStateTimer()

```
void resetExpandedStateTimer ( ) [virtual]
```

Resets the expanded state timer. The SlideMenu will animate to the COLLAPSED state after a number of ticks (set with setExpandedStateTimeout(..)). This method resets this timer.

7.196.2.22 setAnimationDuration()

Parameters

duration The animation duration.

7.196.2.23 setAnimationEasingEquation()

Parameters

animationEasingEquation	The animation easing equation.
-------------------------	--------------------------------

7.196.2.24 setExpandDirection()

Parameters

newExpandDirection	The new expand direction.
--------------------	---------------------------

7.196.2.25 setExpandedStateTimeout()

Sets the expanded state timeout in ticks. The SlideMenu will animate to the COLLAPSED state when this number of ticks has been executed while the SlideMenu is in the EXPANDED state. The timer can be reset with the reset← ExpandedStateTimer method.

timeout	The timeout in ticks.
unicoul	THE UNICOUL IN UCKS.

7.196.2.26 setHiddenPixelsWhenExpanded()

Parameters

```
hiddenPixels The hidden pixels.
```

7.196.2.27 setState()

Parameters

newState The new state of the SlideMenu.
--

7.196.2.28 setStateChangedAnimationEndedCallback()

```
\label{local_continuous_continuous} $$\operatorname{void} \ \operatorname{setStateChangedAnimationEndedCallback} \ ($$\operatorname{GenericCallback} < \operatorname{const} \ \operatorname{SlideMenu} \ \& \ > \& \ \operatorname{\it callback} \ ) \ \ [\operatorname{virtual}] $$
```

Parameters

```
callback The callback.
```

7.196.2.29 setStateChangedCallback()

Parameters

```
callback The callback.
```

7.196.2.30 setup() [1/2]

Setup the SlideMenu by positioning the stateChangeButton next to background image relative to the expand direction and center it in the other dimension. The width and height of the SlideMenu will be automatically set to span

both elements. Default values are: expandedStateTimeout = 200, visiblePixelsWhenCollapsed = 0, hiddenPixels↔ WhenExpanded = 0, animationDuration = 10, animationEquation = cubicEaseInOut.

Parameters

newExpandDirection	The new expand direction.
backgroundBMP	The background bitmap.
stateChangeButtonBMP	The state change button bitmap.
stateChangeButtonPressedBMP	The state change button pressed bitmap.

Setup method for the SlideMenu. Positioning of the background image and the stateChangeButton is done by stating the X and Y coordinates for the elements (relative to the SlideMenu). The width and height of the SlideMenu will be automatically set to span both elements. Default values are: expandedStateTimeout = 200, visiblePixelsWhen \leftarrow Collapsed = 0, hiddenPixelsWhenExpanded = 0, animationDuration = 10, animationEquation = cubicEaseInOut.

Parameters

newExpandDirection	The new expand direction.
backgroundBMP	The background bitmap.
stateChangeButtonBMP	The state change button bitmap.
stateChangeButtonPressedBMP	The state change button pressed bitmap.
backgroundX	The background x coordinate.
backgroundY	The background y coordinate.
stateChangeButtonX	The state change button x coordinate.
stateChangeButtonY	The state change button y coordinate.

7.196.2.32 setVisiblePixelsWhenCollapsed()

	diameters	
ĺ	visiblePixels	The visible pixels.

7.196.2.33 stateChangeButtonClickedHandler()

Parameters

button	The state change button.
--------	--------------------------

7.197 Slider Class Reference

A slider is a graphical element with which the user may set a value by moving an indicator or by clicking the slider.

```
#include <touchgfx/containers/Slider.hpp>
```

Public Member Functions

• Slider ()

Default constructor.

virtual ∼Slider ()

Destructor.

void setBitmaps (const Bitmap &sliderBackground, const Bitmap &sliderBackgroundSelected, const Bitmap &indicator)

Sets all the bitmaps for the Slider.

 void setBitmaps (const Bitmapld sliderBackground, const Bitmapld sliderBackgroundSelected, const Bitmapld indicator)

Sets all the bitmaps for the Slider.

void setStartValueCallback (GenericCallback< const Slider &, int > &callback)

Associates an action to be performed when an interaction (drag or click) with the slider is initiated.

void setStopValueCallback (GenericCallback< const Slider &, int > &callback)

Associates an action to be performed when an interaction with the slider ends (i.e. drag/click).

void setNewValueCallback (GenericCallback< const Slider &, int > &callback)

Associates an action to be performed when the slider changes its value.

virtual void setupHorizontalSlider (uint16_t backgroundX, uint16_t backgroundY, uint16_t indicatorY, uint16_t indicatorMinX, uint16_t indicatorMaxX)

Sets up the slider in horizontal mode.

virtual void setupVerticalSlider (uint16_t backgroundX, uint16_t backgroundY, uint16_t indicatorX, uint16_t indicatorMinY, uint16_t indicatorMaxY)

Sets up the slider in vertical mode.

virtual uint16_t getIndicatorMin () const

Gets indicator minimum.

• virtual uint16_t getIndicatorMax () const

Gets indicator maximum.

• virtual void setValueRange (int minValue, int maxValue, int newValue)

Sets the value range of the slider.

virtual uint16_t getMinValue () const

Gets the minimum value.

virtual uint16_t getMaxValue () const

Gets the maximum value.

virtual void setValueRange (int minValue, int maxValue)

Sets the value range of the slider.

virtual void setValue (int value)

Places the indicator at the specified value.

• int getValue ()

Gets the current value represented by the indicator.

virtual uint16_t getType () const

For GUI testing only.

Protected Types

enum SliderOrientation { HORIZONTAL, VERTICAL }

Values that represent slider orientations.

Protected Member Functions

virtual void handleClickEvent (const ClickEvent &evt)

Updates the indicators position.

virtual void handleDragEvent (const DragEvent &evt)

Updates the indicators position.

virtual void updateIndicatorPosition (int16_t position)

Updates the indicator position described by position.

virtual int16_t valueToPosition (int value) const

Translate a value in the value range to a corresponding indicator position.

virtual int positionToValue (int16 t position) const

Translate a position in the indicator position range to the corresponding value.

virtual uint16_t getIndicatorRadius () const

Gets the indicator radius.

· virtual int getIndicatorPositionRangeSize () const

Gets the indicator position range size.

· virtual int getValueRangeSize () const

Gets the value range size.

Protected Attributes

· SliderOrientation sliderOrientation

The slider orientation.

· int currentValue

The current value represented of the slider.

· int valueRangeMin

The value range min.

· int valueRangeMax

The value range max.

· Image background

The background image.

· Image backgroundSelected

The backgroundSelected image.

Image indicator

The indicator image.

· Container backgroundSelectedViewPort

The backgroundSelected view port. Controls the visible part of the backgroundSelected image.

• int16_t indicatorMinPosition

The minimum position of the indicator (either x coordinate in horizontal mode or y coordinate in vertical mode)

int16_t indicatorMaxPosition

The maximum position of the indicator (either x coordinate in horizontal mode or y coordinate in vertical mode)

GenericCallback
 const Slider &, int > * startValueCallback

The start value callback (called when an interaction with the indicator is initiated)

GenericCallback
 const Slider &, int > * stopValueCallback

The stop value callback (called when an interaction with the indicator ends)

GenericCallback< const Slider &, int > * newValueCallback

The new value callback (called when the indicator is moved)

Additional Inherited Members

7.197.1 Detailed Description

A slider is a graphical element with which the user may set a value by moving an indicator or by clicking the slider. The slider can operate in horizontal or vertical mode.

The slider has two bitmaps. One bitmap is used on one side of the indicator. The other is used on the other side. They can be used in indicating the part of the slider value range that is currently selected.

The slider operates on an integer value range that can be set by the user.

7.197.2 Constructor & Destructor Documentation

```
7.197.2.1 Slider()
```

```
Slider ()
```

Default constructor. Set the value range to default 0-100.

7.197.3 Member Function Documentation

7.197.3.1 getIndicatorMax()

```
uint16_t getIndicatorMax ( ) const [inline], [virtual]
```

Gets indicator maximum previous set using setupHorizontalSlider or setupVerticalSlider.

Returns

The calculated indicator maximum.

See also

setupHorizontalSlider setupVerticalSlider getIndicatorMin

7.197.3.2 getIndicatorMin()

```
uint16_t getIndicatorMin ( ) const [inline], [virtual]
```

Gets indicator minimum previous set using setupHorizontalSlider or setupVerticalSlider.

Returns

The indicator minimum.

See also

setupHorizontalSlider setupVerticalSlider getIndicatorMax

7.197.3.3 getIndicatorPositionRangeSize()

```
int getIndicatorPositionRangeSize ( ) const [protected], [virtual]
```

Returns

The indicator position range size.

7.197.3.4 getIndicatorRadius()

```
uint16_t getIndicatorRadius ( ) const [protected], [virtual]
```

Returns

The the indicator radius.

7.197.3.5 getMaxValue()

```
uint16_t getMaxValue ( ) const [inline], [virtual]
```

Gets the maximum value previously set using setValueRange.

Returns

The maximum value.

See also

setValueRange getMinValue

```
7.197.3.6 getMinValue()
uint16_t getMinValue ( ) const [inline], [virtual]
Gets the minimum value previously set using setValueRange.
Returns
     The minimum value.
See also
     setValueRange
     getMaxValue
7.197.3.7 getType()
uint16_t getType ( ) const [inline], [virtual]
For GUI testing only. Returns type of this drawable.
Returns
     TYPE BUTTON.
Reimplemented from Container.
7.197.3.8 getValue()
int getValue ( ) [inline]
Gets the current value represented by the indicator.
Returns
     The current value.
7.197.3.9 getValueRangeSize()
int getValueRangeSize ( ) const [protected], [virtual]
Returns
     The value range size.
7.197.3.10 handleClickEvent()
```

const ClickEvent & evt) [protected], [virtual]

void handleClickEvent (

Parameters

```
evt The event.
```

Reimplemented from Drawable.

7.197.3.11 handleDragEvent()

Parameters

evt	The event.
-----	------------

Reimplemented from Drawable.

7.197.3.12 positionToValue()

Translate a position (x coordinate in horizontal mode and y in vertical mode) in the indicator position range to the corresponding value in the value range.

Parameters

```
position The position.
```

Returns

The value that corresponds to the coordinate.

7.197.3.13 setBitmaps() [1/2]

Sets all the bitmaps for the Slider. The slider show the sliderBackgroundSelected bitmap in the region of the slider that is selected, that is the area to the left of the indicator for a horizontal slider and below the indicator for a vertical slider. To ignore this effect just add the same bitmap for both the sliderBackground and the sliderBackground Selected.

sliderBackground	The slider background with the slider range unselected.
sliderBackgroundSelected	The slider background with the slider range selected.
indicator	The indicator.

7.197.3.14 setBitmaps() [2/2]

Sets all the bitmaps for the Slider. The slider show the sliderBackgroundSelected bitmap in the region of the slider that is selected, that is the area to the left of the indicator for a horizontal slider and below the indicator for a vertical slider. To ignore this effect just add the same bitmap for both the sliderBackground and the sliderBackground Selected.

Parameters

sliderBackground	The slider background.
sliderBackgroundSelected	The slider background selected.
indicator	The indicator.

7.197.3.15 setNewValueCallback()

Associates an action to be performed when the slider changes its value.

Parameters

callback	The callback to be executed. The callback will be given a reference to the Slider and the current	
	value of the slider.	

See also

GenericCallback

7.197.3.16 setStartValueCallback()

Associates an action to be performed when an interaction (drag or click) with the slider is initiated.

Parameters

callback	The callback to be executed. The callback will be given a reference to the Slider and the current
	value of the slider at interaction start.

See also

GenericCallback

7.197.3.17 setStopValueCallback()

Associates an action to be performed when an interaction with the slider ends (i.e. drag/click).

Parameters

callback	The callback to be executed. The callback will be given a reference to the Slider and the current
	value of the slider at interaction end.

See also

GenericCallback

7.197.3.18 setupHorizontalSlider()

```
void setupHorizontalSlider (
          uint16_t backgroundX,
          uint16_t backgroundY,
          uint16_t indicatorY,
          uint16_t indicatorMinX,
          uint16_t indicatorMaxX ) [virtual]
```

Sets up the slider in horizontal mode with the range going from the left to right.

Places the backgrounds and the indicator inside the Slider container. It is possible to place the end points of the indicator outside the background image if it needs to go beyond the boundaries of the background. The width and height of the Slider will be adjusted appropriately so that both the background and the indicator will be fully visible in both the minimum and maximum indicator positions.

Note that the x and y position of the Slider will either be the left/top of the background or the left/top of the indicator in its minimum x coordinate.

Calls setValue with the current value (default 0) and triggers the newSliderValue callback.

Parameters

backgroundX	The background x coordinate inside the slider.	
backgroundY	The background y coordinate inside the slider.	
indicatorY	The indicator y coordinate inside the slider.	
indicatorMinX	The indicator minimum x coordinate inside the slider. This is the position used when the slider	
	is at its minimum value. Must be less than indicatorMaxX.	
indicatorMaxX		
	slider is at its maximum value. Must be greater than indicatorMinX.	

7.197.3.19 setupVerticalSlider()

```
uint16_t indicatorMaxY ) [virtual]
```

Sets up the slider in vertical mode with the range going from the bottom to top.

Places the backgrounds and the indicator inside the Slider container. It is possible to place the end points of the indicator outside the background image if it needs to go beyond the boundaries of the background. The width and height of the Slider will be adjusted appropriately so that both the background and the indicator will be fully visible in both the minimum and maximum indicator positions.

Note that the x and y position of the Slider will either be the left/top of the background or the left/top of the indicator in its minimum y coordinate.

Calls setValue with the current value (default 0) and triggers the newSliderValue callback.

Parameters

backgroundX	The background x coordinate inside the slider.	
backgroundY	The background y coordinate inside the slider.	
indicatorX	The indicator x coordinate inside the slider.	
indicatorMinY	The indicator minimum y coordinate inside the slider. This is the position used when the slider	
	is at its maximum value. Must be less than indicatorMaxX.	
indicatorMaxY	The indicator maximum y coordinate inside the slider. This is the position used when the	
	slider is at its minimum value. Must be greater than indicatorMinX.	

7.197.3.20 setValue()

Places the indicator at the specified value relative to the specified value range. Values beyond the value range will be rounded to the min/max value in the value range.

Note that the value update triggers a newSliderValue callback just as a drag or click does.

Note that if the value range is larger than the number of pixels specified for the indicator min and max some values will not be represented by the slider and thus is not possible to set with this method. In this case the value will be rounded to the nearest value that is represented in the current setting.

Parameters

value	The value.
-------	------------

7.197.3.21 setValueRange() [1/2]

```
void setValueRange (
    int minValue,
    int maxValue,
    int newValue ) [virtual]
```

Sets the value range of the slider. Values accepted and returned by the slider will be in this range.

The slider will set its value to the specified new value.

Note that if the range is larger than the number of pixels specified for the indicator min and max some values will not be represented by the slider.

Parameters

minValue	The minimum value. Must be less than maxValue.	
maxValue	The maximum value. Must be greater than minValue.	
newValue	The new value.	

7.197.3.22 setValueRange() [2/2]

Sets the value range of the slider. Values accepted and returned by the slider will be in this range.

The slider will set its value to the current value or round to minValue or maxValue if the current value is outside the new range.

Note that if the range is larger than the number of pixels specified for the indicator min and max some values will not be represented by the slider.

Parameters

minValue	The minimum value. Must be less than maxValue.
maxValue	The maximum value. Must be greater than minValue.

7.197.3.23 updateIndicatorPosition()

Updates the indicator position described by position. Calls the newSliderValueCallback with the new value.

Parameters

position	The position (x coordinate in horizontal mode and y coordinate in vertical mode).
----------	---

7.197.3.24 valueToPosition()

Translate a value in the value range to a position in the indicator position range (x coordinate in horizontal mode and y in vertical mode).

Parameters

•	i didiliotoro		
ſ	value	The value.	

Returns

The coordinate that corresponds to the value.

7.198 SlideTransition < templateDirection > Class Template Reference

A Transition that slides from one screen to the next.

```
#include <touchgfx/transitions/SlideTransition.hpp>
```

Public Member Functions

• SlideTransition (const uint8 t transitionSteps=20)

Constructor.

virtual ∼SlideTransition ()

Destructor.

virtual void handleTickEvent ()

Handles the tick event when transitioning.

virtual void tearDown ()

Tear down.

· virtual void init ()

Initializes this object.

Protected Member Functions

• virtual void initMoveDrawable (Drawable &d)

Moves the Drawable to its initial position.

virtual void tickMoveDrawable (Drawable &d)

Moves the Drawable.

Protected Attributes

· SnapshotWidget snapshot

The SnapshotWidget that is moved when transitioning.

SnapshotWidget * snapshotPtr

Pointer pointing to the snapshot used in this transition. The snapshot pointer.

7.198.1 Detailed Description

```
template < Direction template Direction > class touchgfx::Slide Transition < template Direction >
```

A Transition that slides from one screen to the next. It does so by moving a SnapShotWidget with a snapshot of the Screen transitioning away from, and by moving the contents of Screen transitioning to.

Template Parameters

•	
templateDirection	Type of the template direction.

See also

Transition

7.198.2 Constructor & Destructor Documentation

7.198.2.1 SlideTransition()

Constructor.

Parameters

transitionSteps | Number of steps in the transition animation.

7.198.2.2 ~SlideTransition()

```
~SlideTransition () [inline], [virtual]
```

Destructor.

7.198.3 Member Function Documentation

7.198.3.1 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

Handles the tick event when transitioning. It moves the contents of the Screen's container and a SnapshotWidget with a snapshot of the previous Screen. The direction of the transition determines the direction the contents of the container and the SnapshotWidget moves.

Reimplemented from Transition.

7.198.3.2 init()

```
void init ( ) [inline], [virtual]
```

Initializes this object.

See also

Transition::init()

Reimplemented from Transition.

7.198.3.3 initMoveDrawable()

Moves the **Drawable** to its initial position.

Parameters

in	d	The Drawable to move.
----	---	-----------------------

7.198.3.4 tearDown()

```
void tearDown ( ) [inline], [virtual]
```

Tear down.

See also

Transition::teadDown()

Reimplemented from Transition.

7.198.3.5 tickMoveDrawable()

Moves the Drawable.

Parameters

in d The Dra	wable to move.
--------------	----------------

7.199 Snapper < T > Class Template Reference

A mix-in that will make class T draggable and able to snap to a position.

```
#include <touchgfx/mixins/Snapper.hpp>
```

Public Member Functions

• Snapper ()

Default constructor.

virtual ∼Snapper ()

Destructor.

virtual void handleDragEvent (const DragEvent &evt)

Called when dragging the Snapper.

virtual void handleClickEvent (const ClickEvent &evt)

Handles the click events when the Snapper is clicked.

void setSnapPosition (int16_t x, int16_t y)

Sets the position the Snapper should snap to.

void setDragAction (GenericCallback< const DragEvent & > &callback)

Associates an action to be performed when the Snapper is dragged.

void setSnappedAction (GenericCallback<> &callback)

Associates an action to be performed when the Snapper is snapped.

7.199.1 Detailed Description

```
template < class T > class touchgfx::Snapper < T >
```

A mix-in that will make class T draggable and able to snap to a position when a drag operation has ended. The mix-in is able to perform callbacks when the snapper gets dragged and when the Snapper snaps to its snap position.

Template Parameters

```
T specifies the type to enable the Snap behavior to.
```

See also

```
Draggable<T>
```

7.199.2 Constructor & Destructor Documentation

```
7.199.2.1 Snapper()
```

```
Snapper ( ) [inline]
```

Default constructor.

```
7.199.2.2 \sim Snapper()
```

```
~Snapper ( ) [inline], [virtual]
```

Destructor.

7.199.3 Member Function Documentation

7.199.3.1 handleClickEvent()

Handles the click events when the Snapper is clicked. It saves its current position as the snap position if the Snapper is pressed. This happens when the drag operation starts.

The snapper will then move to the snap position when the click is released. This happens when the drag operation ends.

Parameters

```
evt The click event.
```

7.199.3.2 handleDragEvent()

Called when dragging the Snapper. It will delegate the event if a GenericCallback is set with setDragAction.

Parameters

```
evt The drag event.
```

Reimplemented from Draggable < T >.

7.199.3.3 setDragAction()

Associates an action to be performed when the Snapper is dragged.

Parameters

callback	The callback will be executed with the DragEvent.

See also

GenericCallback

7.199.3.4 setSnappedAction()

Associates an action to be performed when the Snapper is snapped.

Parameters

in	callback	The callback to be executed on snap.
----	----------	--------------------------------------

See also

GenericCallback

7.199.3.5 setSnapPosition()

Sets the position the Snapper should snap to. This position will be overridden with the Snappers current position when the Snapper is pressed.

Parameters

Х	The x coordinate.
У	The y coordinate.

7.200 SnapshotWidget Class Reference

A widget that is able to make a snapshot of the area the SnapshotWidget covers.

```
#include <touchgfx/widgets/SnapshotWidget.hpp>
```

Public Member Functions

• SnapshotWidget ()

Default constructor.

virtual ∼SnapshotWidget ()

Destructor.

· virtual void draw (const Rect &invalidatedArea) const

Draws the SnapshotWidget.

• virtual Rect getSolidRect () const

Gets solid rectangle.

virtual void makeSnapshot ()

Makes a snapshot of the area the SnapshotWidget currently covers.

virtual void makeSnapshot (const Bitmapld bmp)

Makes a snapshot of the area the SnapshotWidget currently to a bitmap.

void setAlpha (const uint8_t a)

Sets the alpha value.

• uint8_t getAlpha () const

Gets the current alpha value.

• virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

· Bitmapld bitmapld

BitmapId where copy is stored s copied to.

• uint8_t alpha

The alpha with which to draw this snapshot.

Additional Inherited Members

7.200.1 Detailed Description

A widget that is able to make a snapshot of the area the SnapshotWidget covers. The SnapshotWidget will show the snapshot captured when it is drawn. Note: The snapshot must be taken from a byte aligned position. On BPP=4, this means on even positions, x=0, 2, 4, 8,... On BPP=2, this means on positions, x=0, 4, 8, 12,... On BPP=1, this means on positions, x=0, 8, 16,...

See also

Widget

7.200.2 Constructor & Destructor Documentation

```
7.200.2.1 SnapshotWidget()
```

```
SnapshotWidget ( )
```

Default constructor.

```
7.200.2.2 ~SnapshotWidget()
```

```
~SnapshotWidget ( ) [virtual]
```

Destructor.

7.200.3 Member Function Documentation

Draws the SnapshotWidget. It supports partial drawing, so it only redraws the area described by invalidatedArea.

Parameters

```
invalidatedArea The rectangle to draw, with coordinates relative to this drawable.
```

Implements Drawable.

```
7.200.3.2 getAlpha()
```

```
uint8_t getAlpha ( ) const [inline]
```

Gets the current alpha value.

Returns

The alpha value.

7.200.3.3 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets solid rectangle.

Returns

The solid rectangle.

Implements Drawable.

7.200.3.4 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE SNAPSHOTWIDGET.

Reimplemented from Widget.

```
7.200.3.5 makeSnapshot() [1/2]
```

```
void makeSnapshot ( ) [virtual]
```

Makes a snapshot of the area the SnapshotWidget currently covers. This area is defined by setting the dimensions and the position of the SnapshotWidget. The snapshot is stored in Animation Storage.

```
7.200.3.6 makeSnapshot() [2/2]
```

Makes a snapshot of the area the SnapshotWidget currently covers. This area is defined by setting the dimensions and the position of the SnapshotWidget. The snapshot is stored in the provided dynamic bitmap.

Parameters

bmp The target dynamic bitmap.

7.200.3.7 setAlpha()

```
void setAlpha (
```

```
const uint8_t a ) [inline]
```

Sets the alpha value.

Parameters

a The alpha value.

7.201 LCD::StringVisuals Struct Reference

The visual elements when writing a string.

#include <touchqfx/lcd/LCD.hpp>

Public Member Functions

• StringVisuals ()

Construct an empty StringVisuals object.

Construct a StringVisual object for rendering text.

Public Attributes

const Font * font

The font to use.

Alignment alignment

The alignment to use. Default is LEFT.

· TextDirection textDirection

The direction to use. Default is LTR.

· TextRotation rotation

Orientation (rotation) of the text. Default is TEXT_ROTATE_0.

· colortype color

RGB color value. Default is 0 (black).

• int16_t linespace

Line space in pixels for multiline strings. Default is 0.

• uint8_t alpha

8-bit alpha value. Default is 255 (solid).

uint8_t indentation

Indentation of text inside rectangle. Text will start this far from the left/right edge.

• WideTextAction wideTextAction

What to do with wide text lines.

7.201.1 Detailed Description

The visual elements when writing a string.

7.201.2 Constructor & Destructor Documentation

```
7.201.2.1 StringVisuals() [1/2]
StringVisuals ( ) [inline]
Construct an empty StringVisuals object.
```

7.201.2.2 StringVisuals() [2/2]

Parameters

font	The Font with which to draw the text.	
color	The color with which to draw the text.	
alpha	Alpha blending. Default value is 255 (solid)	
alignment	How to align the text.	
linespace	Line space in pixels between each line, in case the text contains newline characters.	
rotation	How to rotate the text.	
textDirection	The text direction.	
indentation	The indentation of the text from the left and right of the text area rectangle.	
wideTextAction	What to do with lines longer than the width of the TextArea.	

7.202 SwipeContainer Class Reference

A swipe container.

#include <touchgfx/containers/SwipeContainer.hpp>

Public Member Functions

virtual void handleTickEvent ()

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

· virtual void handleClickEvent (const touchgfx::ClickEvent &evt)

Defines the event handler interface for ClickEvents.

virtual void handleDragEvent (const touchgfx::DragEvent &evt)

Defines the event handler interface for DragEvents.

virtual void handleGestureEvent (const touchgfx::GestureEvent &evt)

Defines the event handler interface for GestureEvents.

• virtual void add (touchgfx::Drawable &page)

Adds a page to the container.

virtual void remove (Drawable &page)

Removes the page from the container.

virtual void setSwipeCutoff (uint16_t cutoff)

Set the swipe cutoff.

void setPageIndicatorXY (int16_t x, int16_t y)

Sets the x and y position of the page indicator.

void setPageIndicatorXYWithCenteredX (int16_t x, int16_t y)

Sets the x and y position of the page indicator.

void setPageIndicatorBitmaps (const touchgfx::Bitmap &normalPage, const touchgfx::Bitmap &highlighted
 —
 Page)

Sets the bitmaps that are used by the page indicator.

void setEndSwipeElasticWidth (uint16_t width)

When dragging either one of the end pages a part of the background will become visible until the user stop dragging and the end page swipes back to its position. The width of this area is set by this method.

• uint8_t getNumberOfPages ()

Gets number of pages.

void setSelectedPage (uint8_t pageIndex)

Sets the selected page.

Additional Inherited Members

7.202.1 Detailed Description

See also

touchgfx::Container.

7.202.2 Member Function Documentation

Adds a page to the container.

Parameters

in,out page T	he page to add.
---------------	-----------------

Note

All pages must have the same width and height.

Reimplemented from Container.

```
7.202.2.2 getNumberOfPages()
```

```
uint8_t getNumberOfPages ( ) [inline]
```

Gets number of pages.

Returns

The number of pages.

7.202.2.3 handleClickEvent()

Defines the event handler interface for ClickEvents. The default implementation ignores the event. The event is only received if the drawable is touchable.

Parameters

```
evt The ClickEvent received from the HAL.
```

Reimplemented from Drawable.

7.202.2.4 handleDragEvent()

Defines the event handler interface for DragEvents. The event is only received if the drawable is touchable.

Parameters

```
evt The DragEvent received from the HAL.
```

Reimplemented from Drawable.

7.202.2.5 handleGestureEvent()

Defines the event handler interface for GestureEvents. The default implementation ignores the event. The event is only received if the drawable is touchable.

Parameters

```
evt The GestureEvent received from the HAL.
```

Reimplemented from Drawable.

7.202.2.6 handleTickEvent()

```
virtual void handleTickEvent ( ) [virtual]
```

Called periodically by the framework if the Drawable instance has subscribed to timer ticks.

See also

Application::registerTimerWidget

Reimplemented from Drawable.

7.202.2.7 remove()

Removes the page from the container.

Parameters

in,out	page	The page to remove.
--------	------	---------------------

Note

This is safe to call even if page is not a page (in which case nothing happens).

Reimplemented from Container.

7.202.2.8 setEndSwipeElasticWidth()

Parameters

width	The width in pixels.

7.202.2.9 setPageIndicatorBitmaps()

Sets the bitmaps that are used by the page indicator.

Parameters

normalPage	The normal page.
highlightedPage	The highlighted page.

7.202.2.10 setPageIndicatorXY()

```
{\tt void \ setPageIndicatorXY} \ (
```

```
int16_t x,
int16_t y )
```

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.202.2.11 setPageIndicatorXYWithCenteredX()

Sets the x and y position of the page indicator. The value specified as x will be the center coordinate of the page indicators.

Parameters

Χ	The center x coordinate.
У	The y coordinate.

Note

This method should not be used until all pages have been added, the setPageIndicatorBitmaps has been called and the page indicator therefore has the correct width.

7.202.2.12 setSelectedPage()

Sets the selected page.

Parameters

pageIndex	Zero-based index of the page.	Range from 0 to numberOfPages-1.
-----------	-------------------------------	----------------------------------

7.202.2.13 setSwipeCutoff()

Set the swipe cutoff which indicates how far you should drag a page before it results in a page change.

Parameters

cutoff	The cutoff in pixels.

7.203 TextArea Class Reference

This widget is capable of showing a text area on the screen.

#include <touchgfx/widgets/TextArea.hpp>

Public Member Functions

• TextArea ()

Default constructor.

virtual Rect getSolidRect () const

Gets solid rectangle.

void setColor (colortype color)

Sets the color of the text.

colortype getColor () const

Gets the color of the text.

void setAlpha (uint8_t alpha)

Sets the alpha value of the text.

• uint8_t getAlpha () const

Gets the alpha value of the text.

virtual void setBaselineY (int16_t baselineY)

Adjusts the TextArea y coordinate to place the text at the specified baseline.

virtual void setXBaselineY (int16_t x, int16_t baselineY)

Adjusts the TextArea y coordinate to place the text at the specified baseline.

void setLinespacing (int16_t space)

Sets the line spacing of the TextArea.

int16_t getLinespacing () const

Gets the line spacing of the TextArea.

void setIndentation (uint8_t indent)

Sets the indentation for the text.

• uint8_t getIndentation ()

Gets the indentation.

virtual int16_t getTextHeight ()

Gets the total height needed by the text.

virtual uint16_t getTextWidth () const

Gets the width in pixels of the current associated text in the current selected language.

· virtual void draw (const Rect &area) const

Draws the text.

void setTypedText (TypedText t)

Sets the TypedText of the text area.

TypedText getTypedText () const

Gets the TypedText of the text area.

void setRotation (const TextRotation rotation)

Sets rotation of the text in the TextArea.

TextRotation getRotation () const

Gets rotation of the text in the TextArea.

void resizeToCurrentText ()

Sets the dimensions of the TextArea.

void resizeToCurrentTextWithAlignment ()

Sets the dimensions of the TextArea.

void resizeHeightToCurrentText ()

Sets the height of the TextArea.

· void setWideTextAction (WideTextAction action)

Sets wide text action.

· WideTextAction getWideTextAction () const

Gets wide text action.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

TypedText typedText

The TypedText to display.

colortype color

The color to use.

• int16_t linespace

The line spacing to use, in pixels, in case the text contains newlines.

· uint8 t alpha

The alpha to use.

· uint8 t indentation

The indentation of the text inside the text area.

TextRotation rotation

The text rotation to use.

WideTextAction wideTextAction

What to do if the text is wider than the text area.

Additional Inherited Members

7.203.1 Detailed Description

This widget is capable of showing a text area on the screen. A TextArea can display a TypedText. Optional configuration include text color.

Example text_example shows how to use a TextArea.

Note

A TextArea just holds a pointer to the text displayed. The developer must ensure that the pointer remains valid when drawing.

See also

TypedText for information about text

TextAreaWithOneWildcard,

TextAreaWithTwoWildcards for displaying texts containing wildcards.

7.203.2 Constructor & Destructor Documentation

7.203.2.1 TextArea()

TextArea () [inline]

Create an empty TextArea. Default color is black.

7.203.3 Member Function Documentation

Draws the text. Called automatically.

Parameters

area	The invalidated area.
------	-----------------------

Implements Drawable.

 $Reimplemented \ in \ TextAreaWithTwoWildcards, \ and \ TextAreaWithOneWildcard.$

```
7.203.3.2 getAlpha()
uint8_t getAlpha ( ) const [inline]
```

Gets the alpha value of the text.

Returns

The alpha value. 255 = completely solid. 0 = invisible.

```
7.203.3.3 getColor()
```

```
colortype getColor ( ) const [inline]
```

Gets the color of the text.

Returns

The color to used for drawing the text.

```
7.203.3.4 getIndentation()
```

```
uint8_t getIndentation ( ) [inline]
```

Gets the indentation.

Returns

The indentation.

See also

setIndetation

7.203.3.5 getLinespacing()

```
int16_t getLinespacing ( ) const [inline]
```

Gets the line spacing of the TextArea.

Returns

The line spacing.

7.203.3.6 getRotation()

```
TextRotation getRotation ( ) const [inline]
```

Gets rotation of the text in the TextArea.

Returns

The rotation of the text.

7.203.3.7 getSolidRect()

```
Rect getSolidRect ( ) const [inline], [virtual]
```

Gets solid rectangle.

Returns

the largest solid rectangle for this widget. For a TextArea, this is an empty area.

Implements Drawable.

7.203.3.8 getTextHeight()

```
int16_t getTextHeight ( ) [virtual]
```

Gets the total height needed by the text, taking number of lines and line spacing into consideration.

Returns

the total height needed by the text.

Reimplemented in TextAreaWithTwoWildcards, and TextAreaWithOneWildcard.

7.203.3.9 getTextWidth()

```
uint16_t getTextWidth ( ) const [virtual]
```

Gets the width in pixels of the current associated text in the current selected language. In case of multi-lined text the width of the widest line is returned.

Returns

The width in pixels of the current text.

Reimplemented in TextAreaWithTwoWildcards, and TextAreaWithOneWildcard.

```
7.203.3.10 getType()
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TEXTAREA.

Reimplemented from Widget.

Reimplemented in TextAreaWithTwoWildcards, and TextAreaWithOneWildcard.

```
7.203.3.11 getTypedText()
```

```
TypedText getTypedText ( ) const [inline]
```

Gets the TypedText of the text area.

Returns

The currently used TypedText.

7.203.3.12 getWideTextAction()

```
WideTextAction getWideTextAction ( ) const [inline]
```

Gets wide text action preciously set using setWideTextAction.

Returns

current WideTextAction setting.

See also

setWideTextAction

7.203.3.13 resizeHeightToCurrentText()

```
void resizeHeightToCurrentText ( )
```

Sets the height of the TextArea to match the height of the current associated text for the current selected language. This is especially useful for texts with WordWrap enabled.

Please note that if the current text rotation is either 90 or 270 degrees, the width of the text area will be set and not the height, as the text is rotated.

See also

resizeToCurrentText setWordWrap setRotation

7.203.3.14 resizeToCurrentText()

```
void resizeToCurrentText ( )
```

Sets the dimensions of the TextArea to match the width and height of the current associated text for the current selected language.

Please note that if the current text rotation is either 90 or 270 degrees, the width of the text area will be set to the height of the text and vice versa, as the text is rotated.

See also

```
setRotation
resizeHeightToCurrentText
```

7.203.3.15 resizeToCurrentTextWithAlignment()

```
void resizeToCurrentTextWithAlignment ( )
```

Sets the dimensions of the TextArea to match the width and height of the current associated text for the current selected language.

When setting the width, the position of the TextArea might be changed in order to keep the text centered or right aligned.

Please note that if the current text rotation is either 90 or 270 degrees, the width of the text area will be set to the height of the text and vice versa, as the text is rotated.

See also

```
setRotation
resizeHeightToCurrentText
```

7.203.3.16 setAlpha()

Sets the alpha value of the text.

Parameters

alnha	The alpha value. 255 = completely solid. 0 = invisible.
aipiia	The diplia value. 200 – completely 30lid. 0 – invisible.

7.203.3.17 setBaselineY()

Adjusts the text areas y coordinate so the text will have its baseline at the specified value. The placements is relative to the specified TypedText so if this changes you have to set the baseline again. Note that setTypedText must be called prior to setting the baseline.

Parameters

baselineY The y coordinate of the baseline.

7.203.3.18 setColor()

Sets the color of the text.

Parameters

color The color to use.

7.203.3.19 setIndentation()

Sets the indentation for the text. This is very useful when a font is an italic font where letters such as "j" and "g" extend a lot to the left under the previous characters. if a line starts with a "j" or "g" this letter would either have to be pushed to the right to be able to see all of it, e.g. using spaces which would ruin a multi line text which is left aligned. This could be solved by changing a textarea.setPosition(50,50,100,100) to textarea.setPosition(45,50,110, 100) followed by a textarea.setIndentation(5). Characters that do not extend to the left under the previous characters will be drawn in the same position in either case, but "j" and "g" will be aligned with other lines.

The function getMaxPixelsLeft() will give you the maximum number of pixels any glyph in the font extends to the left.

Parameters

indent The indentation from left (when left aligned text) and right (when right aligned text).

See also

getMaxPixelsLeft

7.203.3.20 setLinespacing()

Sets the line spacing of the TextArea.

Parameters

space	The line spacing of use in the TextArea.
-------	--

7.203.3.21 setRotation()

Sets rotation of the text in the TextArea. The value TEXT_ROTATE_0 is the default for normal text. The value TEXT_ROTATE_90 will rotate the text clockwise, thus writing from the top of the display and down. Similarly TE \leftarrow XT_ROTATE_180 and TEXT_ROTATE_270 is further rotate 90 degrees clockwise.

Parameters

```
rotation The rotation of the text.
```

7.203.3.22 setTypedText()

```
void setTypedText ( {\tt TypedText}\ t\ )
```

Sets the TypedText of the text area. If no prior size has been set the TextArea will be resized to fit the new TypedText.

Parameters

t The TypedText for this widget to display.

7.203.3.23 setWideTextAction()

Sets wide text action. Defines what to do if a line of text is wider than the text area. Default action is WIDE_TEXT—NONE which means that text lines are only broken if there is a newline in the text.

If wrapping is enabled and the text would occupy more lines than the size of the TextArea, the last line will get an ellipsis to signal that some text is missing. The character used for ellipsis is taken from the text spreadsheet.

Parameters

action The action to perform for wide lines o	of text.
---	----------

See also

WideTextAction getWideTextAction resizeHeightToCurrentText

7.203.3.24 setXBaselineY()

Adjusts the text areas y coordinate so the text will have its baseline at the specified value. The placements is relative to the specified TypedText so if this changes you have to set the baseline again. Note that setTypedText must be called prior to setting the baseline. The specified x coordinate will be used as the x coordinate of the TextArea.

Parameters

Х	The x coordinate of the TextArea.
baselineY	The y coordinate of the baseline.

7.204 TextAreaWithOneWildcard Class Reference

TextArea with one wildcard.

```
#include <touchgfx/widgets/TextAreaWithWildcard.hpp>
```

Public Member Functions

• TextAreaWithOneWildcard ()

Default constructor.

virtual int16_t getTextHeight ()

Gets text height.

virtual void draw (const Rect &area) const

Draws TextArea and its text.

void setWildcard (const Unicode::UnicodeChar *value)

Sets the wildcard in the text.

const Unicode::UnicodeChar * getWildcard () const

Gets the wildcard in the text.

• virtual uint16_t getTextWidth () const

Gets the width in pixels of the current associated text.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

• const Unicode::UnicodeChar * wildcard

Pointer to the wildcard string. Must be zero-terminated.

Additional Inherited Members

7.204.1 Detailed Description

TextArea with one wildcard. The format string (i.e. the text pointer set in TextArea::setText) is expected to contain a wildcard s.

See also

TextAreaWithWildcardBase

7.204.2 Constructor & Destructor Documentation

7.204.2.1 TextAreaWithOneWildcard()

```
TextAreaWithOneWildcard ( ) [inline]
```

Create an empty text area.

Note

No text can be displayed until a font is set. Default color is black.

7.204.3 Member Function Documentation

```
7.204.3.1 draw()
void draw (
```

Draws TextArea and its text if a Font is set and the TypedText associated with the TextArea is valid.

const Rect & area) const [inline], [virtual]

Parameters

area	The invalidated area.
arca	The invalidated area.

Reimplemented from TextArea.

7.204.3.2 getTextHeight()

```
int16_t getTextHeight ( ) [inline], [virtual]
```

Gets text height.

Returns

The text height.

Reimplemented from TextArea.

7.204.3.3 getTextWidth()

```
uint16_t getTextWidth ( ) const [inline], [virtual]
```

Gets the width in pixels of the current associated text in the current selected language. In case of multi-lined text the width of the widest line is returned.

Returns

The width in pixels of the current text.

Reimplemented from TextArea.

```
7.204.3.4 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TEXTAREAWITHONEWILDCARD.

Reimplemented from TextArea.

```
7.204.3.5 getWildcard()
```

```
const Unicode::UnicodeChar * getWildcard ( ) const [inline]
```

Gets the wildcard in the text.

Returns

The wildcard used in the text.

7.204.3.6 setWildcard()

Sets the wildcard in the text. Must be a zero-terminated Unicode Char array.

Parameters

value A pointer to the UnicodeChar to set the wildcard to.

7.205 TextAreaWithTwoWildcards Class Reference

TextArea with two wildcards.

```
#include <touchgfx/widgets/TextAreaWithWildcard.hpp>
```

Public Member Functions

• TextAreaWithTwoWildcards ()

Default constructor.

• virtual int16_t getTextHeight ()

Gets text height.

virtual void draw (const Rect &area) const

Draws TextArea and its text.

void setWildcard1 (const Unicode::UnicodeChar *value)

Sets the first wildcard in the text.

• const Unicode::UnicodeChar * getWildcard1 () const

Gets the first wildcard in the text.

void setWildcard2 (const Unicode::UnicodeChar *value)

Sets the second wildcard in the text.

const Unicode::UnicodeChar * getWildcard2 () const

Gets the second wildcard in the text.

• virtual uint16_t getTextWidth () const

Gets the width in pixels of the current associated text.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

const Unicode::UnicodeChar * wc1

Pointer to the first wildcard string. Must be zero-terminated.

const Unicode::UnicodeChar * wc2

Pointer to the second wildcard string. Must be zero-terminated.

Additional Inherited Members

7.205.1 Detailed Description

TextArea with two wildcards. The format string (i.e. the text pointer set in TextArea::setText) is expected to contain two wildcards s.

See also

TextAreaWithWildcardBase

7.205.2 Constructor & Destructor Documentation

7.205.2.1 TextAreaWithTwoWildcards()

TextAreaWithTwoWildcards () [inline]

Create an empty text area.

Note

No text can be displayed until a font is set. Default color is black.

7.205.3 Member Function Documentation

7.205.3.1 draw()

Draws TextArea and its text if a Font is set and the TypedText associated with the TextArea is valid.

Parameters

area The invalidated are	a.
--------------------------	----

Reimplemented from TextArea.

7.205.3.2 getTextHeight()

```
int16_t getTextHeight ( ) [inline], [virtual]
```

Gets text height.

Returns

The text height.

Reimplemented from TextArea.

7.205.3.3 getTextWidth()

```
uint16_t getTextWidth ( ) const [inline], [virtual]
```

Gets the width in pixels of the current associated text in the current selected language. In case of multi-lined text the width of the widest line is returned.

Returns

The width in pixels of the current text.

Reimplemented from TextArea.

7.205.3.4 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TEXTAREAWITHTWOWILDCARDS.

Reimplemented from TextArea.

7.205.3.5 getWildcard1()

```
const Unicode::UnicodeChar * getWildcard1 ( ) const [inline]
```

Gets the first wildcard in the text.

Returns

The first wildcard from a TextArea with two wildcards.

7.205.3.6 getWildcard2()

```
const Unicode::UnicodeChar * getWildcard2 ( ) const [inline]
```

Gets the second wildcard in the text.

Returns

The second wildcard from a TextArea with two wildcards.

7.205.3.7 setWildcard1()

Sets the first wildcard in the text. Must be a zero-terminated UnicodeChar array.

Parameters

value A pointer to the UnicodeChar to set the wildcard to.

7.205.3.8 setWildcard2()

Sets the second wildcard in the text. Must be a zero-terminated UnicodeChar array.

Parameters

value A pointer to the UnicodeChar to set the wildcard to.

7.206 TextAreaWithWildcardBase Class Reference

Base class for TextAreas displaying texts with one or more wildcards.

#include <touchgfx/widgets/TextAreaWithWildcard.hpp>

Public Member Functions

• TextAreaWithWildcardBase ()

Create an empty text area.

• int16_t calculateTextHeight (const Unicode::UnicodeChar *format,...) const

Gets the total height needed by the text.

Additional Inherited Members

7.206.1 Detailed Description

Base class for TextAreas displaying texts with one or more wildcards.

See also

TextAreaWithOneWildcard TextAreaWithTwoWildcards

7.206.2 Constructor & Destructor Documentation

7.206.2.1 TextAreaWithWildcardBase()

```
TextAreaWithWildcardBase ( ) [inline]
```

Create an empty text area.

Note

No text can be displayed until a font is set. Default color is black.

7.206.3 Member Function Documentation

7.206.3.1 calculateTextHeight()

Gets the total height needed by the text. Determined by number of lines and linespace. The number of wildcards in the text should match the number of values for the wildcards.

Parameters

	format	t The text containing s wildcards.	
Variable arguments providing additional inform		Variable arguments providing additional information.	

Returns

the total height needed by the text.

7.207 TextButtonStyle < T > Class Template Reference

A text button style.

#include <touchgfx/containers/buttons/TextButtonStyle.hpp>

Public Member Functions

• TextButtonStyle ()

Default constructor.

virtual ~TextButtonStyle ()

Destructor.

void setText (TypedText t)

Sets a text.

void setTextX (int16_t x)

Sets text x coordinate.

void setTextY (int16_t y)

Sets text y coordinate.

void setTextXY (int16_t x, int16_t y)

Sets text xy.

void setTextPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets text position.

void setTextRotation (TextRotation rotation)

Sets text rotation.

void setTextColors (colortype newColorReleased, colortype newColorPressed)

Sets text colors.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

· virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

TextArea text

The text.

· colortype colorReleased

The color released.

colortype colorPressed

The color pressed.

7.207.1 Detailed Description

template < class T > class touchgfx::TextButtonStyle < T >

An text button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show a text in one of two colors depending on the state of the button (pressed or released).

The TextButtonStyle does not set the size of the enclosing container (normally AbstractButtonContainer). The size must be set manually.

To get a background behind the text, use TextButtonStyle together with e.g. ImageButtonStyle: TextButton \leftarrow Style<ImageButtonStyle<ClickButtonTrigger>> myButton;

The position of the text can be adjusted with setTextXY (default is centered).

Template Parameters

```
T Generic type parameter. Typically a AbstractButtonContainer subclass.
```

See also

AbstractButtonContainer

7.207.2 Member Function Documentation

Parameters

```
t A TypedText to process.
```

7.207.2.2 setTextColors()

Parameters

newColorReleased	The new color released.	
newColorPressed	The new color pressed.	

7.207.2.3 setTextPosition()

```
void setTextPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [inline]
```

Parameters

	•
Y	The x coordinate

Parameters

У	The y coordinate.
width	The width of the text.
height	The height of the text.

7.207.2.4 setTextRotation()

Parameters

7.207.2.5 setTextX()

```
void setTextX (
          int16_t x ) [inline]
```

Parameters

7.207.2.6 setTextXY()

Parameters

Х	The x coordinate.
У	The y coordinate.

7.207.2.7 setTextY()

```
void setTextY ( int16\_t \ y \ ) \quad [inline]
```

Parameters

У	The y coordinate.
---	-------------------

7.208 TextProgress Class Reference

A text progress.

#include <touchgfx/containers/progress_indicators/TextProgress.hpp>

Public Member Functions

• TextProgress ()

Default constructor.

virtual ∼TextProgress ()

Destructor.

• virtual void setProgressIndicatorPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the position and dimension of the text progress indicator.

virtual void setTypedText (const TypedText &t)

Sets the typed text.

virtual TypedText getTypedText () const

Gets the typed text.

virtual void setColor (colortype color)

Sets the color.

virtual colortype getColor () const

Gets the color.

virtual void setAlpha (uint8_t alpha)

Sets the alpha.

virtual uint8_t getAlpha () const

Gets the alpha.

virtual void setValue (int value)

Sets a value.

• virtual void setNumberOfDecimals (uint16_t numberOfDecimals)

Sets number of decimals.

virtual uint16_t getNumberOfDecimals () const

Gets number of decimals.

Protected Attributes

TextAreaWithOneWildcard textArea

The text area.

• Unicode::UnicodeChar textBuffer [9]

Room for 100.0000.

· uint16 t decimals

The number of decimals.

Additional Inherited Members

7.208.1 Detailed Description

A text progress will display progress as a number with a given number of decimals.

Note

The implementation does not use floating point variables to calculate the progress.

7.208.2 Constructor & Destructor Documentation

```
7.208.2.1 TextProgress()
TextProgress ( )
Default constructor.
7.208.2.2 \simTextProgress()
~TextProgress ( ) [virtual]
Destructor.
7.208.3 Member Function Documentation
7.208.3.1 getAlpha()
uint8_t getAlpha ( ) const [virtual]
Gets the alpha of the text area.
Returns
     The alpha.
See also
     setAlpha
     TextArea::getAlpha
7.208.3.2 getColor()
colortype getColor ( ) const [virtual]
Gets the color of the text in the used text area.
Returns
     The color.
7.208.3.3 getNumberOfDecimals()
uint16_t getNumberOfDecimals ( ) const [virtual]
Gets number of decimals.
```

Returns

The number of decimals.

See also

setNumberOfDecimals

```
7.208.3.4 getTypedText()
```

```
TypedText getTypedText ( ) const [virtual]
```

Gets the typed text.

Returns

The typed text.

See also

setTypedText

7.208.3.5 setAlpha()

Sets the alpha of the text area.

Parameters

```
alpha The alpha.
```

See also

getAlpha

TextArea::setAlpha

7.208.3.6 setColor()

Sets the color of the text in the used text area.

Parameters

color	The color.

See also

 ${\sf getColor}$

TextArea::setColor

7.208.3.7 setNumberOfDecimals()

Sets number of decimals when displaying progress.

Parameters

|--|

See also

getNumberOfDecimals

7.208.3.8 setProgressIndicatorPosition()

```
void setProgressIndicatorPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the position and dimension of the text progress indicator relative to the background image.

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the text progress indicator.
height	The height of the text progress indicator.

Reimplemented from AbstractProgressIndicator.

7.208.3.9 setTypedText()

Sets the typed text. The text should have one wildcard and could for example "< progress>%".

Parameters

t The TypedText to process.

See also

getTypedText

7.208.3.10 setValue()

```
\begin{tabular}{ll} \begin{tabular}{ll} virtual & void & setValue & ( & \\ & int & value & ) & [virtual] \end{tabular}
```

Sets the current value in the range (min..max) set by setRange(). Values lower than min are mapped to min, values higher than max are mapped to max.

Parameters

Reimplemented from AbstractProgressIndicator.

7.209 TextProvider Class Reference

The TextProvider is used in drawing basic and wildcard strings.

```
#include <touchgfx/TextProvider.hpp>
```

Public Member Functions

• TextProvider ()

Default constructor.

• void initialize (const Unicode::UnicodeChar *stringFormat, va_list pArg, const uint16_t *gsubTable=0)

Initializes the TextProvider.

• Unicode::UnicodeChar getNextChar ()

Gets the next character.

Unicode::UnicodeChar getNextLigature (TextDirection direction)

Gets the next ligature.

- Unicode::UnicodeChar getNextLigature (TextDirection direction, const Font *font, const GlyphNode *&glyph)

 Gets the next ligature.
- Unicode::UnicodeChar getNextLigature (TextDirection direction, const Font *font, const GlyphNode *&glyph, const uint8_t *&pixelData, uint8_t &bitsPerPixel)

Gets the next ligature.

Static Public Attributes

static const uint32_t MAX_32BIT_INTEGER_DIGITS = 33U
 Max number of digits used for the text representation of a 32 bit integer.

7.209.1 Detailed Description

The TextProvider is used in drawing basic and wildcard strings. The TextProvider enables wildcard expansion of the string at the time it is written to the LCD.

It provides printf formatted text strings one character at the time, without the need for a user provided buffer to store the text string.

7.209.2 Constructor & Destructor Documentation

7.209.2.1 TextProvider()

```
TextProvider ( )
```

Empty constructor. The user must call initialize() before characters can be provided.

7.209.3 Member Function Documentation

7.209.3.1 getNextChar()

```
Unicode::UnicodeChar getNextChar ( )
```

Gets the next character. For Arabic and Thai, it is important to use the getNextLigature instead.

Returns

The next character of the expanded string or 0 if end of string is reached.

See also

TextProvider::getNextLigature()

7.209.3.2 getNextLigature() [1/3]

Gets the next ligature. For most languages this is the same as getNextChar() but eg. Arabic has different ligatures for each character. Thai character placement might also depend on previous characters. It is recommended to use getNextLigature with font and glyph parameters to ensure coming glyphs in a text are placed correctly.

Note

Functions getNextLigature() and getNextChar() will advance through the same buffer and mixing the use of those functions is not recommended and may cause undesired results. Instead create two TextProviders and user getNextChar() on one and getNextLigature() on the other.

Parameters

```
direction The direction.
```

Returns

The next character of the expanded string or 0 if end of string is reached.

See also

TextProvider::getNextChar()

7.209.3.3 getNextLigature() [2/3]

Gets the next ligature. For most languages this is the same as getNextChar() but eg. Arabic has different ligatures for each character.

Also gets a glyph for the ligature in a font. For non-Thai unicodes, this is identical to using Font::getGlyph(), but for Thai characters where diacritics glyphs are not always placed at the same relative position, an adjusted GlyphNode will be generated with correct relative X/Y coordinates.

Note

Functions getNextLigature() and getNextChar() will advance through the same buffer and mixing the use of those functions is not recommended and may cause undesired results. Instead create two TextProviders and user getNextChar() on one and getNextLigature() on the other.

Parameters

	direction	The direction.
	font	The font.
out	glyph	The glyph.

Returns

The next character of the expanded string or 0 if end of string i reached.

See also

```
TextProvider::getNextChar() Font::getGlyph
```

7.209.3.4 getNextLigature() [3/3]

Gets the next ligature. For most languages this is the same as getNextChar() but eg. Arabic has different ligatures for each character.

Also gets a glyph for the ligature in a font. For non-Thai unicodes, this is identical to using Font::getGlyph(), but for Thai characters where diacritics glyphs are not always placed at the same relative position, an adjusted GlyphNode will be generated with correct relative X/Y coordinates.

Furthermore a pointer to the glyph data and the bit depth of the font are returned in parameters.

Note

Functions getNextLigature() and getNextChar() will advance through the same buffer and mixing the use of those functions is not recommended and may cause undesired results. Instead create two TextProviders and user getNextChar() on one and getNextLigature() on the other.

Parameters

	direction	The direction.
	font	The font.
out	glyph	The glyph.
out	pixelData	Information describing the pixel.
out	bitsPerPixel	The bits per pixel.

Returns

The next character of the expanded string or 0 if end of string is reached.

See also

TextProvider::getNextChar() Font::getGlyph

7.209.3.5 initialize()

Initializes the TextProvider. Each '\2' character in the format is replaced by one UnicodeChar* argument from pArg.

Parameters

stringFormat	The string to format.
pArg	Format arguments in the form of a va_list.
gsubTable	Pointer to GSUB table with unicode substitution rules.

7.210 Texts Class Reference

Class for setting language and getting texts.

```
#include <touchgfx/Texts.hpp>
```

Public Member Functions

• const Unicode::UnicodeChar * getText (TypedTextId id) const Get text in the set language.

Static Public Member Functions

• static void setLanguage (Languageld id)

Sets the current language for texts.

• static LanguageId getLanguage ()

Gets the current language.

• static void setTranslation (touchgfx::LanguageId id, const void *translation)

Adds or replaces a translation.

7.210.1 Detailed Description

Class for setting language and getting texts. The language set will determine which texts will be used.

7.210.2 Member Function Documentation

```
7.210.2.1 getLanguage()
static LanguageId getLanguage ( ) [inline], [static]
Gets the current language.
```

Returns

The id of the language.

7.210.2.2 getText()

Get text in the set language.

Parameters

```
id The id of the text to lookup.
```

Returns

The text.

7.210.2.3 setLanguage()

```
static void setLanguage ( {\tt LanguageId} \ id \ ) \quad [{\tt static}]
```

Sets the current language for texts.

Parameters

id The id of the language.

7.210.2.4 setTranslation()

Adds or replaces a translation. This function allows an application to add a translation at runtime.

Parameters

id	The id of the language to add or replace.
translation	A pointer to the translation in flash or RAM.

7.211 TextureMapper Class Reference

The TextureMapper class is a widget capable of drawing a transformed image.

```
#include <touchgfx/widgets/TextureMapper.hpp>
```

Public Types

• enum RenderingAlgorithm { NEAREST_NEIGHBOR, BILINEAR_INTERPOLATION } Rendering algorithms of the image.

Public Member Functions

• TextureMapper ()

Default constructor.

• virtual \sim TextureMapper ()

Destructor.

virtual void setBitmap (const Bitmap &bmp)

Sets the bitmap for the image.

· Bitmap getBitmap () const

Gets the bitmap for the image.

· virtual void draw (const Rect &invalidatedArea) const

Draws the given invalidated area.

virtual Rect getSolidRect () const

Gets solid rectangle.

virtual void setRenderingAlgorithm (RenderingAlgorithm algorithm)

Sets the algorithm to be used.

virtual RenderingAlgorithm getRenderingAlgorithm () const

Gets the algorithm used when rendering.

void setAlpha (uint8 t a)

Sets the global alpha blending value.

uint8_t getAlpha () const

Gets the current alpha value.

virtual void updateAngles (float xAngle, float yAngle, float zAngle)

Updates the angles of the image.

virtual void updateXAngle (float xAngle)

Updates the x coordinate angle described by xAngle.

virtual void updateYAngle (float yAngle)

Updates the y coordinate angle described by yAngle.

virtual void updateZAngle (float zAngle)

Updates the z coordinate angle described by zAngle.

· virtual float getXAngle () const

Get x angle.

· virtual float getYAngle () const

Get y angle.

• virtual float getZAngle () const

Get z angle.

virtual void setScale (float scale)

Sets the scale of the image.

• virtual float getScale () const

Gets the scale.

virtual void setOrigo (float x, float y, float z)

Sets the transformation origo.

virtual void setOrigo (float x, float y)

Sets the transformation origo.

virtual float getOrigoX () const

Gets transformation origo x coordinate.

virtual float getOrigoY () const

Gets transformation origo y coordinate.

virtual float getOrigoZ () const

Gets transformation origo z coordinate.

virtual void setCamera (float x, float y)

Sets the camera coordinate.

virtual float getCameraX () const

Gets camera x coordinate.

virtual float getCameraY () const

Gets camera y coordinate.

virtual void setCameraDistance (float d)

Sets camera distance.

• virtual float getCameraDistance () const

Gets camera distance.

virtual void setBitmapPosition (float x, float y)

Sets bitmap position.

virtual void setBitmapPosition (int x, int y)

Sets bitmap position.

virtual float getBitmapPositionX () const

Gets bitmap position x coordinate.

· virtual float getBitmapPositionY () const

Gets bitmap position y coordinate.

virtual float getX0 () const

Get X0 coordinate.

virtual float getX1 () const

Get X1 coordinate.

· virtual float getX2 () const

Get X2 coordinate.

· virtual float getX3 () const

Get X3 coordinate.

· virtual float getY0 () const

Get Y0 coordinate.

virtual float getY1 () const

Get Y1 coordinate.

virtual float getY2 () const

Get Y2 coordinate.

· virtual float getY3 () const

Get Y3 coordinate.

virtual float getZ0 () const

Get Z0 coordinate.

virtual float getZ1 () const

Get Z1 coordinate.

• virtual float getZ2 () const

Get Z2 coordinate.

• virtual float getZ3 () const

Get Z3 coordinate.

virtual uint16_t getType () const

For GUI testing only.

Protected Member Functions

• void applyTransformation ()

Applies the transformation.

· Rect getBoundingRect () const

Gets bounding rectangle.

void drawTriangle (const Rect &invalidatedArea, uint16_t *fb, const float *triangleXs, const float *triangleYs, const float *triangleZs, const float *triangleUs, const float *triangleVs) const

Draw triangle.

· RenderingVariant lookupRenderVariant () const

Returns the rendering variant based on the bitmap format, alpha value and rendering algorithm.

Protected Attributes

• RenderingAlgorithm currentRenderingAlgorithm

The current rendering algorithm.

Bitmap bitmap

The bitmap to render.

uint8_t alpha

An alpha value that is applied to the entire image.

float xBitmapPosition

The bitmap position x.

float yBitmapPosition

The bitmap position y.

· float xAngle

The angle x.

float yAngle

The angle y.

float zAngle

The angle z.

· float scale

The scale.

· float xOrigo

The origo x coordinate.

float yOrigo

The origo y coordinate.

· float zOrigo

The origo z coordinate.

float xCamera

The camera x coordinate.

float yCamera

The camera y coordinate.

· float cameraDistance

The camera distance.

float imageX0

The coordinate for the image points.

• float imageY0

The coordinate for the image points.

· float imageZ0

The coordinate for the image points.

· float imageX1

The coordinate for the image points.

float imageY1

The coordinate for the image points.

float imageZ1

The coordinate for the image points.

float imageX2

The coordinate for the image points.

float imageY2

The coordinate for the image points.

float imageZ2

The coordinate for the image points.

· float imageX3

The coordinate for the image points.

float imageY3

The coordinate for the image points.

float imageZ3

The coordinate for the image points.

• uint16 t subDivisionSize

The size of the affine sub divisions.

Static Protected Attributes

• static const int MINIMAL_CAMERA_DISTANCE = 1

The minimal camera distance.

7.211.1 Detailed Description

The TextureMapper displays a transformed image. The TextureMapper can be used in effects where an image should be rotated in two or three dimensions.

The image can be freely scaled and rotated in three dimensions. The scaling and rotation is done around the adjustable origin. A virtual camera is applied to the rendered image yielding a perspective impression. The amount of perspective impression can be adjusted. The transformed image is clipped according to the dimensions of the TextureMapper. In order to make the image fully visible the TextureMapper should be large enough to accommodate the transformed image.

Note that the drawing of this widget is not trivial and typically has a significant effect on the mcu load. The number of pixels drawn, the presence of global alpha or per pixel alpha inflicts the computation and should be considered.

Note that this widget does not support 1 bit per pixel color depth.

See also

Widget

7.211.2 Member Enumeration Documentation

7.211.2.1 RenderingAlgorithm

enum RenderingAlgorithm

Rendering algorithms of the image.

NEAREST_NEIGHBOR: Fast algorithm with medium image quality. Good for fast animations. (Default)

BILINEAR_INTERPOLATION: Slower algorithm but better image quality.

7.211.3 Constructor & Destructor Documentation

7.211.3.1 \sim TextureMapper()

```
~TextureMapper ( ) [virtual]
```

Destructor.

7.211.4 Member Function Documentation

7.211.4.1 applyTransformation()

```
void applyTransformation ( ) [protected]
```

Transform the bitmap using the supplied origo, scale, rotation and camera. This method is called by all the methods that manipulate origo, scale, rotation and camera.

7.211.4.2 draw()

Draws the given invalidated area. The part of the transformed image inside the invalidatedArea will be drawn.

Parameters

	invalidatedArea	The rectangle to draw, with coordinates relative to this drawable.
--	-----------------	--

See also

Drawable::draw()

Implements Drawable.

7.211.4.3 drawTriangle()

The TextureMapper will draw the transformed bitmap by drawing two triangles. One triangle is created from the points 0,1,2 and the other triangle from the points 1,2,3. The triangle is drawn using the x,y,z values from each point along with the u,v coordinates in the bitmap associated with each point.

Parameters

	invalidatedArea	The invalidated area.
in,out	fb	If non-null, the fb.
	triangleXs	The triangle xs.
	triangleYs	The triangle ys.
	triangleZs	The triangle zs.
	triangleUs	The triangle us.
	triangleVs	The triangle vs.

7.211.4.4 getAlpha()

```
uint8_t getAlpha ( ) const [inline]
```

Gets the current alpha value.

Returns

The current alpha value.

```
See also
```

setAlpha

```
7.211.4.5 getBitmap()
```

```
Bitmap getBitmap ( ) const [inline]
```

Gets the bitmap for the image.

Returns

the bitmap.

7.211.4.6 getBitmapPositionX()

```
float getBitmapPositionX ( ) const [inline], [virtual]
```

Gets bitmap position x coordinate.

Returns

The bitmap position x coordinate.

7.211.4.7 getBitmapPositionY()

```
float getBitmapPositionY ( ) const [inline], [virtual]
```

Gets bitmap position y coordinate.

Returns

The bitmap position y coordinate.

7.211.4.8 getBoundingRect()

```
Rect getBoundingRect ( ) const [protected]
```

Gets bounding rectangle of the transformed bitmap.

Returns

The bounding rectangle.

7.211.4.9 getCameraDistance()

```
float getCameraDistance ( ) const [inline], [virtual]
```

Gets camera distance.

Returns

The camera distance.

7.211.4.10 getCameraX()

```
float getCameraX ( ) const [inline], [virtual]
```

Gets camera x coordinate.

Returns

The camera x coordinate.

7.211.4.11 getCameraY()

```
float getCameraY ( ) const [inline], [virtual]
```

Gets camera y coordinate.

Returns

The camera y coordinate.

7.211.4.12 getOrigoX()

```
float getOrigoX ( ) const [inline], [virtual]
```

Gets transformation origo x coordinate.

Returns

The transformation origo x coordinate.

7.211.4.13 getOrigoY()

```
float getOrigoY ( ) const [inline], [virtual]
```

Gets transformation origo y coordinate.

Returns

The transformation origo y coordinate.

7.211.4.14 getOrigoZ()

```
float getOrigoZ ( ) const [inline], [virtual]
```

Gets transformation origo z coordinate.

Returns

The transformation origo z coordinate.

```
7.211.4.15 getRenderingAlgorithm()
```

```
RenderingAlgorithm getRenderingAlgorithm ( ) const [inline], [virtual]
```

Gets the algorithm used when rendering.

Returns

The algorithm used when rendering.

```
7.211.4.16 getScale()
```

```
float getScale ( ) const [inline], [virtual]
```

Gets the scale.

Returns

The scale.

7.211.4.17 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets solid rectangle.

Returns

largest possible solid rect.

See also

Drawable::getSolidRect()

Implements Drawable.

```
7.211.4.18 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TEXTUREMAPPER.

Reimplemented from Widget.

```
906
7.211.4.19 getX0()
float getX0 ( ) const [inline], [virtual]
Get the x coordinate of the top left corner of the transformed bitmap.
Returns
     The X0 coordinate.
7.211.4.20 getX1()
float getX1 ( ) const [inline], [virtual]
Get the x coordinate of the top right corner of the transformed bitmap.
Returns
     The X1 coordinate.
7.211.4.21 getX2()
float getX2 ( ) const [inline], [virtual]
Get the x coordinate of the bottom right of the transformed bitmap.
Returns
     The X2 coordinate.
7.211.4.22 getX3()
float getX3 ( ) const [inline], [virtual]
Get the x coordinate of the bottom left corner of the transformed bitmap.
Returns
     The X3 coordinate.
```

```
7.211.4.23 getXAngle()
```

float getXAngle () const [inline], [virtual]

Get x angle.

Returns

The x angle.

```
7.211.4.24 getY0()
float getY0 ( ) const [inline], [virtual]
Get the y coordinate of the top left corner of the transformed bitmap.
Returns
     The Y0 coordinate.
7.211.4.25 getY1()
float getY1 ( ) const [inline], [virtual]
Get the y coordinate of the top right corner of the transformed bitmap.
Returns
     The Y1 coordinate.
7.211.4.26 getY2()
float getY2 ( ) const [inline], [virtual]
Get the y coordinate of the bottom right corner of the transformed bitmap.
Returns
     The Y2 coordinate.
7.211.4.27 getY3()
float getY3 ( ) const [inline], [virtual]
Get the y coordinate of the bottom left corner of the transformed bitmap.
Returns
     The Y3 coordinate.
7.211.4.28 getYAngle()
float getYAngle ( ) const [inline], [virtual]
Get y angle.
```

Returns

The y angle.

```
7.211.4.29 getZ0()
float getZ0 ( ) const [inline], [virtual]
Get the z coordinate of the top left corner of the transformed bitmap.
Returns
     The Z0 coordinate.
7.211.4.30 getZ1()
float getZ1 ( ) const [inline], [virtual]
Get the z coordinate of the top right corner of the transformed bitmap.
Returns
     The Z1 coordinate.
7.211.4.31 getZ2()
float getZ2 ( ) const [inline], [virtual]
Get the z coordinate of the bottom right corner of the transformed bitmap.
Returns
     The Z2 coordinate.
7.211.4.32 getZ3()
float getZ3 ( ) const [inline], [virtual]
Get the z coordinate of the bottom left corner of the transformed bitmap.
Returns
     The Z3 coordinate.
```

7.211.4.33 getZAngle()

float getZAngle () const [inline], [virtual]

Get z angle.

Returns

The z angle.

7.211.4.34 lookupRenderVariant()

```
RenderingVariant lookupRenderVariant ( ) const [protected]
```

Returns the rendering variant based on the bitmap format, alpha value and rendering algorithm.

Returns

The Rendering Variant.

7.211.4.35 setAlpha()

Sets the global alpha blending value.

Parameters

```
a new alpha.
```

7.211.4.36 setBitmap()

Sets the bitmap for the image. Note that the width and height of the TextureMapper is set to the size of the image.

Parameters

bmp	The bitmap to be used by the widget.

7.211.4.37 setBitmapPosition() [1/2]

```
void setBitmapPosition ( \label{eq:float} \mbox{float } x, \\ \mbox{float } y \;) \quad \mbox{[inline], [virtual]}
```

Sets the position of the bitmap within the TextureMapper. The bitmap is clipped against the dimensions of the TextureMapper.

Parameters

Х	The x coordinate.
У	The y coordinate.

7.211.4.38 setBitmapPosition() [2/2]

Sets the position of the bitmap within the TextureMapper. The bitmap is clipped against the dimensions of the TextureMapper.

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.211.4.39 setCamera()

```
void setCamera ( \label{eq:float} \mbox{float } x, \\ \mbox{float } y \; ) \; \mbox{[inline], [virtual]}
```

Sets the camera coordinate.

Parameters

X	The x coordinate for the camera.
У	The y coordinate for the camera.

7.211.4.40 setCameraDistance()

```
void setCameraDistance ( \label{eq:condition} \mbox{float } d \; ) \; \; \mbox{[inline], [virtual]}
```

Sets camera distance. Minimal allowed distance is MINIMAL_CAMERA_DISTANCE. Values below will be set to MINIMAL_CAMERA_DISTANCE.

Parameters

```
d The new camera distance.
```

7.211.4.41 setOrigo() [1/2]

```
void setOrigo (
          float x,
          float y,
          float z ) [inline], [virtual]
```

Sets the transformation origo.

Parameters

Х	The x coordinate.
У	The y coordinate.

Parameters

```
z The z coordinate.
```

7.211.4.42 setOrigo() [2/2]

```
void setOrigo ( \label{eq:float x, float y, float y, float y, float y, float y, float y, float float
```

Sets the transformation origo.

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.211.4.43 setRenderingAlgorithm()

Sets the algorithm to be used. Default setting is NEAREST_NEIGHBOR.

Parameters

algorithm	The algorithm to use when rendering.	1
aigoritiiii	incaigontini to asc when rendering.	ı

7.211.4.44 setScale()

Sets the scale of the image.

Parameters

scale	The new scale value.
-------	----------------------

7.211.4.45 updateAngles()

Updates the angles of the image.

Parameters

xAngle	The new x Angle.
yAngle	The new y Angle.
zAngle	The new x Angle.

7.211.4.46 updateXAngle()

Updates the x coordinate angle described by xAngle.

Parameters

xAngle The new x angle.

7.211.4.47 updateYAngle()

Updates the y coordinate angle described by yAngle.

Parameters

yAngle	The new y angle.
--------	------------------

7.211.4.48 updateZAngle()

Updates the z coordinate angle described by zAngle.

Parameters

zAngle The new z angle.

7.212 TextureSurface Struct Reference

A texture source. Contains a pointer to the data and the width and height of the texture. The alpha channel is used in 565 rendering with alpha. The stride is the width used when moving to the next line of the texture.

```
#include <touchgfx/hal/Types.hpp>
```

Public Attributes

const uint16 t * data

The pixel bits or indexe for color in CLUT entries.

const uint8 t * extraData

The alpha channel or clut data.

· int32_t width

The width.

· int32 t height

The height.

· int32_t stride

The stride.

7.213 TiledImage Class Reference

Simple widget capable of showing a tiled bitmap.

```
#include <touchqfx/widgets/TiledImage.hpp>
```

Public Member Functions

TiledImage (const Bitmap &bmp=Bitmap())

Default Constructor.

virtual void setBitmap (const Bitmap &bmp)

Sets the bitmap ID for this TiledImage.

virtual void setOffset (int16_t x, int16_t y)

Sets an offset into the bitmap where the tile drawing should start.

virtual void setXOffset (int16_t x)

Sets x offset into the bitmap where the tile drawing should start.

virtual void setYOffset (int16_t y)

Sets y offset into the bitmap where the tile drawing should start.

virtual void getOffset (int16_t &x, int16_t &y)

Gets the offset into the bitmap where the tile drawing should start.

virtual int16_t getXOffset ()

Get x coordinate offset.

virtual int16_t getYOffset ()

Get y coordinate offset.

virtual void draw (const Rect &invalidatedArea) const

Draws the image.

virtual Rect getSolidRect () const

Gets the largest solid (non-transparent) rectangle.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

int16_t xOffset

The X offset into the bitmap to start drawing.

int16_t yOffset

The Y offset into the bitmap to start drawing.

Additional Inherited Members

7.213.1 Detailed Description

Simple widget capable of showing a tiled bitmap. This means that when TiledImage is larger than the provided Bitmap, the Bitmap is repeated over and over horizontally and vertically. The bitmap can be alpha-blended with the background and have areas of transparency.

See also

Image

7.213.2 Constructor & Destructor Documentation

```
7.213.2.1 TiledImage()
```

Constructs a new Image with a default alpha value of 255 (solid) and a default Bitmap if none is specified.

Parameters

bmp The bitmap to displa	y.
--------------------------	----

7.213.3 Member Function Documentation

```
7.213.3.1 draw()
```

Draws the image. This class supports partial drawing, so only the area described by the rectangle will be drawn.

Parameters

invali	idatedArea	The rectangle to draw, with coordinates relative to this drawable.
--------	------------	--

Reimplemented from Image.

7.213.3.2 getOffset()

```
void getOffset ( int16\_t \ \& \ x, int16\_t \ \& \ y \ ) \quad [virtual]
```

Gets the offset into the bitmap where the tile drawing should start. Please note that the offsets set using setOffset have been normalized.

Parameters

out	Х	The x coordinate offset.
out	у	The y coordinate offset.

See also

getXOffset getYOffset

7.213.3.3 getSolidRect()

```
Rect getSolidRect ( ) const [virtual]
```

Gets the largest solid (non-transparent) rectangle. This value is pre-calculated by the image converter tool.

Returns

The largest solid (non-transparent) rectangle.

Reimplemented from Image.

7.213.3.4 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_IMAGE.

Reimplemented from Image.

7.213.3.5 getXOffset()

```
int16_t getXOffset ( ) [virtual]
```

Returns

The x coordinate offset.

See also

getYOffset getOffset

7.213.3.6 getYOffset()

```
int16_t getYOffset ( ) [virtual]
```

Returns

The y coordinate offset.

See also

getXOffset getOffset

7.213.3.7 setBitmap()

Sets the bitmap ID for this TiledImage. Updates the width and height of this widget to match that of the bitmap.

Parameters

bmp	The bitmap instance.
-----	----------------------

See also

Bitmap

Reimplemented from Image.

7.213.3.8 setOffset()

```
void setOffset ( int16\_t \ x, int16\_t \ y \ ) \quad [virtual]
```

Sets an offset into the bitmap where the tile drawing should start.

Parameters

,	Υ	The x coordinate offset.
J	/	The y coordinate offset.

See also

setXOffset setYOffset

7.213.3.9 setXOffset()

```
void setXOffset (
```

```
int16_t x ) [virtual]
```

Sets x offset into the bitmap where the tile drawing should start.

Parameters

```
x The x coordinate offset.
```

See also

setYOffset setOffset

7.213.3.10 setYOffset()

```
void setYOffset (
                int16_t y ) [virtual]
```

Sets y offset into the bitmap where the tile drawing should start.

Parameters

```
y The y coordinate offset.
```

See also

setXOffset setOffset

7.214 TiledImageButtonStyle < T > Class Template Reference

A tiled image button style.

```
#include <touchgfx/containers/buttons/TiledImageButtonStyle.hpp>
```

Public Member Functions

• TiledImageButtonStyle ()

Default constructor.

- virtual \sim TiledImageButtonStyle ()

Destructor.

virtual void setWidth (int16_t width)

Sets a width.

virtual void setHeight (int16_t height)

Sets a height.

• virtual void setTileBitmaps (const Bitmap &bmpReleased, const Bitmap &bmpPressed)

Sets tile bitmaps.

virtual void setTileOffset (int16_t x, int16_t y)

Sets an offset into the bitmap where the tile drawing should start.

Protected Member Functions

· virtual void handlePressedUpdated ()

Handles the pressed updated.

virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· TiledImage tiledImage

The tiled image.

· Bitmap upTile

The image to display when button is released.

· Bitmap downTile

The image to display when button is pressed.

7.214.1 Detailed Description

```
\label{template} \begin{split} \text{template} \! < \! \text{class T} \! > \\ \text{class touchgfx::TiledImageButtonStyle} \! < \! \text{T} \! > \\ \end{split}
```

An tiled image button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show one of two tiled images depending on the state of the button (pressed or released).

The TiledImageButtonStyle does not set the size of the enclosing container (normally AbstractButtonContainer) to the size of the pressed Bitmap. This can be overridden by calling setWidth/setHeight after setting the bitmaps.

Template Parameters

T Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer

7.214.2 Member Function Documentation

```
7.214.2.1 setHeight()
```

Parameters

height The height.

7.214.2.2 setTileBitmaps()

Parameters

bmpReleased	The bitmap released.
bmpPressed	The bitmap pressed.

7.214.2.3 setTileOffset()

```
void setTileOffset (
          int16_t x,
          int16_t y) [inline], [virtual]
```

Sets an offset into the bitmap where the tile drawing should start.

Parameters

Χ	The x coordinate offset.
У	The y coordinate offset.

7.214.2.4 setWidth()

Parameters

width The width.

7.215 ToggleButton Class Reference

A ToggleButton is a Button specialization that swaps the two bitmaps when clicked.

```
#include <touchgfx/widgets/ToggleButton.hpp>
```

Public Member Functions

• ToggleButton ()

Default constructor.

• virtual void setBitmaps (const Bitmap &bmpReleased, const Bitmap &bmpPressed)

Sets the bitmaps.

• void forceState (bool activeState)

Force the button into a specific state.

• bool getState () const

Gets the state.

• virtual void handleClickEvent (const ClickEvent &event)

Overrides handleClickEvent.

virtual uint16_t getType () const

For GUI testing only.

Protected Attributes

· Bitmap originalPressed

Contains the bitmap that was originally being displayed when button is pressed.

Additional Inherited Members

7.215.1 Detailed Description

A ToggleButton is a Button specialization that swaps the two bitmaps when clicked, such that the previous "pressed" bitmap, now becomes the one displayed when button is not pressed.

See also

Button

7.215.2 Constructor & Destructor Documentation

7.215.2.1 ToggleButton()

```
ToggleButton ( )
```

Default constructor.

7.215.3 Member Function Documentation

7.215.3.1 forceState()

Use this function to force the button in one of the two possible states. If button is forced to the active state, then the pressed bitmap from the last call to setBitmaps becomes the one displayed when button is not pressed.

Parameters

activeState If true, display the bmpPressed bitmap when not pressed. If false display the bmpReleased bitmap.

7.215.3.2 getState()

```
bool getState ( ) const [inline]
```

Gets the state.

Returns

true if state is currently active.

7.215.3.3 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TOGGLEBUTTON.

Reimplemented from Button.

7.215.3.4 handleClickEvent()

Overrides handleClickEvent in order to swap the bitmaps after being clicked.

Parameters

_	
event	The event to handle.
CVCIIL	The event to nandie.

Reimplemented from AbstractButton.

7.215.3.5 setBitmaps()

Sets the bitmaps.

Note

This specific implementation remembers what bitmap was used as pressed, in order to support the ability to force the state.

Parameters

bmpReleased	The bitmap to show in the "normal" state, ie when button is not pressed.
bmpPressed	The bitmap to show when the button is pressed.

See also

Button::setBitmaps

Reimplemented from Button.

7.216 ToggleButtonTrigger Class Reference

A toggle button trigger.

#include <touchgfx/containers/buttons/ToggleButtonTrigger.hpp>

Public Member Functions

• ToggleButtonTrigger ()

Default constructor.

• virtual ~ToggleButtonTrigger ()

Destructor.

· void forceState (bool activeState)

Force the button into a specific state.

void setToggleCanceled (bool isToggleCanceled)

Sets toggle canceled.

• bool getToggleCanceled ()

Gets toggle canceled.

virtual void handleClickEvent (const ClickEvent &event)

Handles the click event described by event.

Protected Attributes

· bool toggleCanceled

True if toggle canceled.

Additional Inherited Members

7.216.1 Detailed Description

A toggle button trigger. This trigger will create a button that reacts on clicks. This means it will call the action when it gets a touch released event.

The ToggleButtonTrigger will stay in pressed state until it is clicked again.

The ToggleButtonTrigger can be combined with one or more of the ButtonStyle classes to create a functional button.

7.216.2 Member Function Documentation

7.216.2.1 forceState()

```
void forceState (
                bool activeState ) [inline]
```

Use this function to force the button in one of the two possible states. If button is forced to the active state, then AbstractButtonContainer will be in a pressed state.

Parameters

activeState If true, the AbstractButtonContainer will appear pressed.

7.216.2.2 getToggleCanceled()

```
bool getToggleCanceled ( ) [inline]
```

Returns

True if it succeeds, false if it fails.

7.216.2.3 handleClickEvent()

Parameters

event	The event.
-------	------------

Reimplemented from Drawable.

7.216.2.4 setToggleCanceled()

Parameters

inTagglaCanaolad	True if is toggle canceled, false if not
is roggiet anceied	i True il is toddie canceled Talse il not

7.217 TouchArea Class Reference

Invisible widget used to capture touch events.

```
#include <touchgfx/widgets/TouchArea.hpp>
```

Public Member Functions

• TouchArea ()

Default constructor.

- virtual void draw (const Rect &invalidatedArea) const
 - A TouchArea will not draw anything.
- virtual void handleDragEvent (const DragEvent &evt)

A TouchArea will not move when dragged.

- virtual void handleClickEvent (const ClickEvent &event)
 - A TouchArea will refine the handling of click events.
- · virtual Rect getSolidRect () const
 - A TouchArea has no solid rectangle.
- void setPressedAction (GenericCallback< const AbstractButton & > &callback)
 - Associates an action to be performed when the TouchArea is pressed.
- virtual uint16_t getType () const
 - For GUI testing only.

Protected Attributes

• GenericCallback< const AbstractButton &> * pressedAction

The action to perform when the TouchArea is clicked.

Additional Inherited Members

7.217.1 Detailed Description

Invisible widget used to capture touch events. The TouchArea consumes drag events without the widget it self moving.

See also

AbstractButton

7.217.2 Constructor & Destructor Documentation

```
7.217.2.1 TouchArea()
```

```
TouchArea ( ) [inline]
```

Default constructor.

7.217.3 Member Function Documentation

```
7.217.3.1 draw()
```

A TouchArea will not draw anything.

Parameters

	invalidatedArea	The region of the toucharea to draw. Ignored.
--	-----------------	---

Implements Drawable.

7.217.3.2 getSolidRect()

```
Rect getSolidRect ( ) const [inline], [virtual]
```

A TouchArea has no solid rectangle.

Returns

an empty rect.

Implements Drawable.

```
7.217.3.3 getType()
```

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_TOUCHAREA.

Reimplemented from AbstractButton.

7.217.3.4 handleClickEvent()

A TouchArea will refine the handling of click events in order to enable the callback to the pressedAction.

Parameters

```
event The event to handle.
```

Reimplemented from AbstractButton.

7.217.3.5 handleDragEvent()

A TouchArea will not move when dragged.

Parameters

evt The event to handle. Ignored.

Reimplemented from Drawable.

7.217.3.6 setPressedAction()

Associates an action to be performed when the TouchArea is pressed.

Parameters

7.218 TouchButtonTrigger Class Reference

A touch button trigger.

```
#include <touchgfx/containers/buttons/TouchButtonTrigger.hpp>
```

Public Member Functions

• TouchButtonTrigger ()

Default constructor.

• virtual \sim TouchButtonTrigger ()

Destructor

virtual void handleClickEvent (const ClickEvent &event)

Handles the click event described by event.

Additional Inherited Members

7.218.1 Detailed Description

A touch button trigger. This trigger will create a button that reacts on touches. This means it will call the action when it gets a touch pressed event.

The TouchButtonTrigger can be combined with one or more of the ButtonStyle classes to create a functional button.

7.218.2 Member Function Documentation

7.218.2.1 handleClickEvent()

Parameters

```
event The event.
```

Reimplemented from Drawable.

7.219 TouchCalibration Class Reference

Calibrates a touch coordinate.

```
#include <touchgfx/transforms/TouchCalibration.hpp>
```

Static Public Member Functions

static void setCalibrationMatrix (const Point *ref, const Point *scr)

Initializes the calibration matrix based on reference and measured values.

static void translatePoint (Point &p)

Translates the specified point using the matrix.

7.219.1 Detailed Description

Class TouchCalibraiton is responsible for translating coordinates (Point) based on matrix of calibration values.

7.219.2 Member Function Documentation

7.219.2.1 setCalibrationMatrix()

Initializes the calibration matrix based on reference and measured values.

Parameters

ref	Pointer to array of three reference points.	
scr	Pointer to array of three measured points.	

7.219.2.2 translatePoint()

Translates the specified point using the matrix. If matrix has not been initialized, p is not modified.

Parameters

in, out p The point to translate.

7.220 TouchController Class Reference

Basic Touch Controller interface.

#include <platform/driver/touch/TouchController.hpp>

Public Member Functions

virtual ~TouchController ()

Destructor.

virtual void init ()=0

Initializes touch controller.

virtual bool sampleTouch (int32_t &x, int32_t &y)=0

Checks whether the touch screen is being touched, and if so, what coordinates.

7.220.1 Detailed Description

Basic Touch Controller interface.

7.220.2 Constructor & Destructor Documentation

```
7.220.2.1 ~TouchController()
  ~TouchController ( ) [inline], [virtual]
Destructor.
```

7.220.3 Member Function Documentation

```
7.220.3.1 init()
void init ( ) [pure virtual]
```

Initializes touch controller.

Implemented in I2CTouchController, SDL2TouchController, SDLTouchController, and NoTouchController.

7.220.3.2 sampleTouch()

Checks whether the touch screen is being touched, and if so, what coordinates.

Parameters

out	Х	The x position of the touch
out	у	The y position of the touch

Returns

True if a touch has been detected, otherwise false.

Implemented in I2CTouchController, NoTouchController, SDL2TouchController, and SDLTouchController.

7.221 Transition Class Reference

The Transition class is the base class for Transitions.

#include <touchgfx/transitions/Transition.hpp>

Public Member Functions

• Transition ()

Default constructor.

virtual ∼Transition ()

Destructor.

virtual void handleTickEvent ()

Called for every tick when transitioning.

• bool isDone () const

Query if the transition is done transitioning.

• virtual void tearDown ()

Tears down the Animation.

• virtual void init ()

Initializes the transition.

virtual void setScreenContainer (Container &cont)

Sets the screen container.

Protected Attributes

• Container * screenContainer

The screen Container of the Screen transitioning to.

• bool done

Flag that indicates when the transition is done. This should be set by implementing classes.

7.221.1 Detailed Description

The Transition class is the base class for Transitions. Implementations of Transition defines what happens when transitioning between Screens, which typically involves visual effects. An example of a transition implementation can be seen in example custom_transition_example. The most basic transition is the NoTransition class that does a transition without any visual effects.

See also

NoTransition SlideTransition

7.221.2 Constructor & Destructor Documentation

7.221.2.1 Transition()

Transition () [inline]

Constructs the Transition.

7.221.2.2 \sim Transition()

```
\simTransition ( ) [inline], [virtual]
```

Destructor.

7.221.3 Member Function Documentation

7.221.3.1 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

Called for every tick when transitioning. Base does nothing.

Reimplemented in CoverTransition < templateDirection >, SlideTransition < templateDirection >, and NoTransition.

7.221.3.2 init()

```
void init ( ) [inline], [virtual]
```

Initializes the transition. Called after the c.tor. when the application changes the transition. Base version does nothing.

Reimplemented in CoverTransition < templateDirection >, and SlideTransition < templateDirection >.

7.221.3.3 isDone()

```
bool isDone ( ) const [inline]
```

Query if the transition is done transitioning. It is the responsibility of the inheriting class to set the underlying done flag.

Returns

True if the transition is done, false otherwise.

7.221.3.4 setScreenContainer()

Sets the screen container. Is used by Screen to enable the transition to access the container.

Parameters

		T
1 n	cont	The container the transition should have access to.

7.221.3.5 tearDown()

```
void tearDown ( ) [inline], [virtual]
```

Tears down the Animation. Called before the d.tor. when the application changes the transition. Base version does nothing.

Reimplemented in CoverTransition < templateDirection >, and SlideTransition < templateDirection >.

7.222 TwoWildcardTextButtonStyle < T > Class Template Reference

A wildcard text button style.

```
#include <TwoWildcardTextButtonStyle.hpp>
```

Public Member Functions

• TwoWildcardTextButtonStyle ()

Default constructor.

virtual ~TwoWildcardTextButtonStyle ()

Destructor.

void setTwoWildcardText (TypedText t)

Sets wildcard text.

void setTwoWildcardTextX (int16_t x)

Sets wildcard text x coordinate.

void setTwoWildcardTextY (int16_t y)

Sets wildcard text y coordinate.

void setTwoWildcardTextXY (int16_t x, int16_t y)

Sets wildcard text xy.

• void setTwoWildcardTextPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets text position.

void setTwoWildcardTextRotation (TextRotation rotation)

Sets wildcard text rotation.

void setWildcardTextBuffer1 (const Unicode::UnicodeChar *value)

Sets the first wildcard in the text.

void setWildcardTextBuffer2 (const Unicode::UnicodeChar *value)

Sets the second wildcard in the text.

void setTwoWildcardTextColors (colortype newColorReleased, colortype newColorPressed)

Sets wild card text colors.

Protected Member Functions

virtual void handlePressedUpdated ()

Handles the pressed updated.

virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

TextAreaWithTwoWildcards twoWildcardText

The wildcard text.

colortype colorReleased

The color released.

· colortype colorPressed

The color pressed.

7.222.1 Detailed Description

```
\label{template} $$ $$ template < class T > $$ class touchgfx::TwoWildcardTextButtonStyle < T > $$
```

An wildcard text button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show a text with a wildcard in one of two colors depending on the state of the button (pressed or released).

The TwoWildcardTextButtonStyle does not set the size of the enclosing container (normally AbstractButton← Container). The size must be set manually.

To get a background behind the text, use TwoWildcardTextButtonStyle together with e.g. ImageButtonStyle: Two↔ WildcardTextButtonStyle<ImageButtonStyle<ClickButtonTrigger> > myButton;

The position of the text can be adjusted with setTwoWildcardTextXY (default is centered).

Template Parameters

T Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer

Template Parameters

T Generic type parameter.

7.222.2 Member Function Documentation

7.222.2.1 setTwoWildcardText()

Parameters

 $t \mid A \text{ TypedText to process.}$

7.222.2.2 setTwoWildcardTextColors()

Parameters

newColorReleased	The new color released.
newColorPressed	The new color pressed.

7.222.2.3 setTwoWildcardTextPosition()

```
void setTwoWildcardTextPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [inline]
```

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the text.
height	The height of the text.

7.222.2.4 setTwoWildcardTextRotation()

Parameters

7.222.2.5 setTwoWildcardTextX()

```
void setTwoWildcardTextX (
          int16_t x ) [inline]
```

Parameters

x The x coordinate.

7.222.2.6 setTwoWildcardTextXY()

```
void setTwoWildcardTextXY (
          int16_t x,
           int16_t y ) [inline]
```

Parameters

Χ	The x coordinate.
У	The y coordinate.

7.222.2.7 setTwoWildcardTextY()

```
void setTwoWildcardTextY (
          int16_t y ) [inline]
```

Parameters

y The y coordinate.

7.222.2.8 setWildcardTextBuffer1()

Sets the first wildcard in the text. Must be a zero-terminated UnicodeChar array.

Parameters

	value	A pointer to the UnicodeChar to set the wildcard to.
--	-------	--

7.222.2.9 setWildcardTextBuffer2()

Sets the second wildcard in the text. Must be a zero-terminated UnicodeChar array.

Parameters

value A pointer to the UnicodeChar to set the wildcard to.

7.223 TypedText Class Reference

TypedText represents text (as in characters) and typography (as in font and alignment).

```
#include <touchgfx/TypedText.hpp>
```

Classes

struct TypedTextData

The data structure for typed texts.

Public Member Functions

TypedText (const TypedTextId id=TYPED_TEXT_INVALID)

Construct a typed text.

TypedTextId getId () const

Gets the id of the typed text.

• bool has ValidId () const

Has the TypedText been set to a proper value.

const Unicode::UnicodeChar * getText () const

Gets the text associated with this TypedText.

const Font * getFont () const

Gets the font associated with this TypedText.

FontId getFontId () const

Gets the font ID associated with this TypedText.

· Alignment getAlignment () const

Gets the alignment associated with this TypedText.

• TextDirection getTextDirection () const

Gets the text direction associated with this TypedText.

Static Public Member Functions

- static void registerTypedTextDatabase (const TypedTextData *data, const Font *const *f, const uint16_t n)

 Registers an array of typed texts.
- static void registerTexts (const Texts *t)

Registers an array of texts.

7.223.1 Detailed Description

TypedText represents text (as in characters) and typography (as in font and alignment). TypedText provides methods for interacting with the text, font and alignment.

Example text example shows how to use TypedText.

See also

TextArea

7.223.2 Constructor & Destructor Documentation

Construct a typed text.

Parameters

id The id of the TypedText.

7.223.3 Member Function Documentation

```
7.223.3.1 getAlignment()

Alignment getAlignment ( ) const [inline]

Gets the alignment associated with this TypedText.

Returns

The alignment.
```

```
7.223.3.2 getFont()
```

```
const Font * getFont ( ) const [inline]
```

Gets the font associated with this TypedText.

Returns

The font.

```
7.223.3.3 getFontId()
```

```
FontId getFontId ( ) const [inline]
```

Gets the font ID associated with this TypedText.

Returns

The font.

```
7.223.3.4 getId()
```

```
TypedTextId getId ( ) const [inline]
```

Gets the id of the typed text.

Returns

The id.

7.223.3.5 getText()

```
const Unicode::UnicodeChar * getText ( ) const [inline]
```

Gets the text associated with this TypedText.

Returns

The text.

7.223.3.6 getTextDirection()

```
TextDirection getTextDirection ( ) const [inline]
```

Gets the text direction associated with this TypedText.

Returns

The alignment.

7.223.3.7 hasValidId()

```
bool hasValidId ( ) const [inline]
```

Has the TypedText been set to a proper value.

Returns

Is the id valid.

7.223.3.8 registerTexts()

Registers an array of texts.

Parameters

```
t The array of texts.
```

7.223.3.9 registerTypedTextDatabase()

Registers an array of typed texts. All typed text instances are bound to this database.

Parameters

data	A reference to the TypedTextData storage array.	
f	The fonts associated with the array.	
n The number of typed texts in the array.		

7.224 TypedText::TypedTextData Struct Reference

The data structure for typed texts.

#include <touchgfx/TypedText.hpp>

Public Attributes

· const unsigned char fontldx

The font associated with the typed text.

· const Alignment alignment: 2

The alignment of the typed text.

· const TextDirection direction: 2

The text direction (LTR,RTL,...) of the typed text.

7.224.1 Detailed Description

The data structure for typed texts.

7.225 UIEventListener Class Reference

This class declares a handler interface for user interface events.

#include <touchgfx/UIEventListener.hpp>

Public Member Functions

· virtual void handleClickEvent (const ClickEvent &event)

This handler is invoked when a mouse click or display touch event has been detected by the system.

virtual void handleDragEvent (const DragEvent &event)

This handler is invoked when a drag event has been detected by the system.

virtual void handleGestureEvent (const GestureEvent &event)

This handler is invoked when a gesture event has been detected by the system.

virtual void handleKeyEvent (uint8 t c)

This handler is invoked when a key (or button) event has been detected by the system.

virtual void handleTickEvent ()

This handler is invoked when a system tick event has been generated.

virtual void handlePendingScreenTransition ()

This handler is invoked when a change screen event is pending.

virtual ∼UIEventListener ()

Destructor.

7.225.1 Detailed Description

This class declares a handler interface for user interface events, i.e. events generated by the users interaction with the device. With the exception of the system timer tick, all other system events, which are not related to the user interface device peripherals (display, keys etc.) are not part of this interface.

7.225.2 Constructor & Destructor Documentation

```
7.225.2.1 ~UIEventListener()
~UIEventListener ( ) [inline], [virtual]
Destructor.
```

7.225.3 Member Function Documentation

7.225.3.1 handleClickEvent()

This handler is invoked when a mouse click or display touch event has been detected by the system.

Parameters

```
event The event data.
```

Reimplemented in Application.

7.225.3.2 handleDragEvent()

This handler is invoked when a drag event has been detected by the system.

Parameters

```
event The event data.
```

Reimplemented in Application.

7.225.3.3 handleGestureEvent()

This handler is invoked when a gesture event has been detected by the system.

Parameters

```
event The event data.
```

Reimplemented in Application.

7.225.3.4 handleKeyEvent()

This handler is invoked when a key (or button) event has been detected by the system.

Parameters

```
c The key or button pressed.
```

Reimplemented in Application.

7.225.3.5 handlePendingScreenTransition()

```
void handlePendingScreenTransition ( ) [inline], [virtual]
```

This handler is invoked when a change screen event is pending.

Reimplemented in Application, and MVPApplication.

7.225.3.6 handleTickEvent()

```
void handleTickEvent ( ) [inline], [virtual]
```

This handler is invoked when a system tick event has been generated. The system tick period is configured in the HAL.

Reimplemented in Application.

7.226 Unicode Class Reference

This class provides simple helper functions for working with 16-bit strings.

```
#include <touchgfx/Unicode.hpp>
```

Public Types

• typedef uint16_t UnicodeChar

Use the UnicodeChar typename when referring to strings.

Static Public Member Functions

static uint16_t strlen (const UnicodeChar *str)

Gets the length of a 0-terminated unicode string.

static uint16_t strlen (const char *str)

Gets the length of a 0-terminated string.

 static uint16_t strncpy (UnicodeChar *RESTRICT dst, const UnicodeChar *RESTRICT src, uint16_t maxchars)

Copy a string to a destination buffer, UnicodeChar to UnicodeChar version.

• static uint16_t strncpy (UnicodeChar *RESTRICT dst, const char *RESTRICT src, uint16_t maxchars)

Copy a string to a destination buffer, char to UnicodeChar version.

• static void itoa (int32_t value, UnicodeChar *buffer, uint16_t bufferSize, int radix)

Integer to ASCII conversion.

static void utoa (uint32 t value, UnicodeChar *buffer, uint16 t bufferSize, int radix)

Integer to ASCII conversion.

static int atoi (const UnicodeChar *s)

String to integer conversion.

static UnicodeChar * snprintf (UnicodeChar *dst, uint16_t dstSize, const UnicodeChar *format,...)

Formats a string and adds null termination.

static UnicodeChar * vsnprintf (UnicodeChar *dst, uint16_t dstSize, const UnicodeChar *format, va_list pArg)

Variant of special format

static UnicodeChar * snprintf (UnicodeChar *dst, uint16_t dstSize, const char *format,...)

Variant of snprintf.

static UnicodeChar * vsnprintf (UnicodeChar *dst, uint16_t dstSize, const char *format, va_list pArg)
 Variant of snprintf.

 static UnicodeChar * snprintfFloats (UnicodeChar *dst, uint16_t dstSize, const UnicodeChar *format, const float *values)

Variant of snprintf for floats only.

 static UnicodeChar * snprintfFloat (UnicodeChar *dst, uint16_t dstSize, const UnicodeChar *format, const float value)

Variant of snprintf.

 static UnicodeChar * snprintfFloats (UnicodeChar *dst, uint16_t dstSize, const char *format, const float *values)

Variant of snprintf for floats only.

- static UnicodeChar * snprintfFloat (UnicodeChar *dst, uint16_t dstSize, const char *format, const float value) Variant of snprintf.
- static int strncmp (const UnicodeChar *RESTRICT str1, const UnicodeChar *RESTRICT str2, uint16_←
 t maxchars)

Compares up to maxchars characters of the string str1 to those of the string str2.

 static int strncmp_ignore_white_spaces (const UnicodeChar *RESTRICT str1, const UnicodeChar *REST← RICT str2, uint16 t maxchars)

Like strncmp except that ignore any spaces in the two strings.

• static uint16_t fromUTF8 (const uint8_t *utf8, UnicodeChar *dst, uint16_t maxchars)

Convert a string from utf8 to unicode.

• static uint16 t toUTF8 (const UnicodeChar *unicode, uint8 t *utf8, uint16 t maxbytes)

Converts a string from unicode to utf8.

Static Public Attributes

· static const UnicodeChar EMPTY [1]

An empty string, which should be used instead of a null-pointer to indicate that the a string has no value.

7.226.1 Member Typedef Documentation

7.226.1.1 UnicodeChar

```
uint16_t UnicodeChar
```

Use the UnicodeChar typename when referring to strings.

7.226.2 Member Function Documentation

String to integer conversion. Starts conversion at the start of the string. Running digits from here are converted.

Parameters

```
s DECIMAL zero-terminated unicode string to convert.
```

Returns

The converted integer value of the string, 0 if the string does not start with a digit.

7.226.2.2 fromUTF8()

Convert a string from utf8 to unicode. The conversion stops if there is no more room in the destination or if the terminating zero character has been converted.

Parameters

		utf8	The UTF8 string.
οι	ut	dst	The destination buffer for the converted string.
		maxchars	The maximum number of chars that the dst array can hold.

Returns

The number of characters successfully converted from utf8 to unicode including the terminating zero.

7.226.2.3 itoa()

```
uint16_t bufferSize,
int radix ) [static]
```

Integer to ASCII conversion.

Parameters

	value	to convert.
out	buffer	to place result in.
	bufferSize Size of buffer (number of 16-bit values).	
	radix	to use (8 for octal, 10 for decimal, 16 for hex)

...) [static]

Formats a string and adds null termination. The string is formatted like when printf is used.

Support formats: %c (element type: char), %s (element type: zero-terminated UnicodeChar list), %u, %i, %d, %o, %x (all these are integers formatted in radix 10, 10, 10, 8, 16 respectively).

The number formats (%u, %i, %d, %o and %x) all support %[0][length]X to specify the size of the generated field (length) and whether the number should be prefixed with zeros (or blanks).

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string.
		The values to insert in the format string.

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintfFloat, snprintfFloats

Variant of snprintf.

Support formats: %c (element type: char), %s (element type: zero-terminated UnicodeChar list), %u, %i, %d, %o, %x (all these are integers formatted in radix 10, 10, 10, 8, 16 respectively).

The number formats (%u, %i, %d, %o and %x) all support

```
%[flags][width][.precision]X
```

Where flags can be:

```
'-': left justify the field (see width).
'+': force sign.
'': insert space if value is positive.
'0': left pad with zeros instead of spaces (see width).
```

Where width is the desired width of the output. If the value is larger, more characters may be generated, but not more than the parameter dstSize. If width is '*' the actual width is read from the parameters passed to this function.

Where precision is the number of number of digits after the decimal point, default is 3. Use "%.f" to not generate any numbers after the decimal point. If precision is '*' the actual precision is read from the parameters passed to this function.

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string.
		The values to insert in the format string.

Returns

pointer to the first element in the buffer where the formatted string is placed.

Note

%f is not supported by this function because floats are converted to doubles when given as parameters in a variable argument list (va_list). Use snprintfFloat or snprintfFloats instead.

Warning

The format string is internally copied from at char* to a UnicodeChar*. This buffer has a limit of 63 characters, so if the format is longer than 63 characters, the caller must do this copying to prevent an assert from triggering:

```
touchgfx::Unicode::UnicodeChar tmpfmt[200];
touchgfx::Unicode::strncpy(tmpfmt, "Very, very, very,
```

See also

snprintfFloat, snprintfFloats

```
uint16_t dstSize,
const UnicodeChar * format,
const float value ) [inline], [static]
```

Variant of snprintf for one float only.

The number format supports only one %f with flags/modifiers. The following is supported:

```
%[flags][width][.precision]f
```

Where flags can be:

```
'-': left justify the field (see width).
'+': force sign.
' ': insert space if value is positive.
'#': insert decimal point even if there are not decimals.
'0': left pad with zeros instead of spaces (see width).
```

Where width is the desired width of the output. If the value is larger, more characters may be generated, but not more than the parameter dstSize.

Where precision is the number of number of digits after the decimal point, default is 3. Use "%.f" to not generate any numbers after the decimal point.

```
Unicode::UnicodeChar buffer[20];
Unicode::snprintfFloat(buffer, 20, "%6.4f", 3.14159f);
// buffer="3.1416" Unicode::snprintfFloat(buffer, 20, "%#6.f", 3.14159f);
// buffer=" 3." Unicode::snprintfFloat(buffer, 20, "%6f", 3.14159f);
// buffer="3.142" Unicode::snprintfFloat(buffer, 20, "%+06.f", 3.14159f);
// buffer="+00003"
```

If more control over the output is needed, see snprintfFloats which can have more than a single "%f" in the string and also supports "*" in place of a number.

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string containing exactly on occurrence of f.
	value	The floating point value to insert for f.

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintf, snprintfFloats

7.226.2.7 snprintfFloat() [2/2]

Variant of snprintf for one float only.

The number format supports only one %f with flags/modifiers. The following is supported:

```
%[flags][width][.precision]f
```

Where flags can be:

```
'-': left justify the field (see width).
'+': force sign.
' ': insert space if value is positive.
'#': insert decimal point even if there are not decimals.
'0': left pad with zeros instead of spaces (see width).
```

Where width is the desired width of the output. If the value is larger, more characters may be generated, but not more than the parameter dstSize.

Where precision is the number of number of digits after the decimal point, default is 3. Use "%.f" to not generate any numbers after the decimal point.

```
Unicode::UnicodeChar buffer[20];
Unicode::snprintfFloat(buffer, 20, "%6.4f", 3.14159f);
// buffer="3.1416"
Unicode::snprintfFloat(buffer, 20, "%#6.f", 3.14159f);
// buffer=" 3."
Unicode::snprintfFloat(buffer, 20, "%6f", 3.14159f);
// buffer="3.142"
Unicode::snprintfFloat(buffer, 20, "%+06.f", 3.14159f);
// buffer="+00003"
```

If more control over the output is needed, see snprintfFloats which can have more than a single "%f" in the string and also supports "*" in place of a number.

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string containing exactly on occurrence of f.
	value	The floating point value to insert for f.

Returns

pointer to the first element in the buffer where the formatted string is placed.

Warning

The format string is internally copied from at char* to a UnicodeChar*. This buffer has a limit of 63 characters, so if the format is longer than 63 characters, the caller must do this copying to prevent an assert from triggering:

```
touchgfx::Unicode::UnicodeChar tmpfmt[200];
touchgfx::Unicode::strncpy(tmpfmt, "Very, very, very,
```

See also

snprintf, snprintfFloats

7.226.2.8 snprintfFloats() [1/2]

Variant of snprintf for floats only.

The format supports several %f with flags/modifiers. The following is supported:

```
%[flags][width][.precision]f
```

Where flags can be:

```
'-': left justify the field (see width).
'+': force sign.
' ': insert space if value is positive
'#': insert decimal point even if there are not decimals
'0': left pad with zeros instead of spaces (see width)
```

Where width is the desired width of the output. If the value is larger, more characters may be generated, but not more than the parameter dstSize. If width is '*' the actual width is read from the list of values passed to this function.

Where precision is the number of number of digits after the decimal point, default is

1. Use "%.f" to not generate any numbers after the decimal point. If precision is '*' the actual precision is read from the list of values passed to this function.

```
float paraml[3] = { 6.0f, 4.0f, 3.14159f };
Unicode::snprintfFloats(buffer, 20, "%*.*f", paraml);
// buffer="3.1416"
float param2[2] = { 3.14159f, -123.4f };
Unicode::snprintfFloats(buffer, 20, "%f %f", param2);
// buffer="3.142 -123.400"
```

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string containing f's.
in	values	The floating point values to insert for f. The number of elements in the array must match the number of f's in the format string.

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintf, snprintfFloat

```
uint16_t dstSize,
const char * format,
const float * values ) [static]
```

Variant of snprintf for floats only.

The format supports several %f with flags/modifiers. The following is supported:

```
%[flags][width][.precision]f
```

Where flags can be:

```
'-': left justify the field (see width).
'+': force sign.
' ': insert space if value is positive.
'#': insert decimal point even if there are not decimals.
'0': left pad with zeros instead of spaces (see width).
```

Where width is the desired width of the output. If the value is larger, more characters may be generated, but not more than the parameter dstSize. If width is '*' the actual width is read from the list of values passed to this function.

Where precision is the number of number of digits after the decimal point, default is

1. Use "%.f" to not generate any numbers after the decimal point. If precision is '*' the actual precision is read from the list of values passed to this function.

```
float paraml[3] = { 6.0f, 4.0f, 3.14159f };
Unicode::snprintfFloats(buffer, 20, "%*.*f", paraml);
// buffer="3.1416"
float param2[2] = { 3.14159f, -123.4f };
Unicode::snprintfFloats(buffer, 20, "%f %f", param2);
// buffer="3.142 -123.400"
```

Parameters

out	dst	Buffer for the formatted string.	
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.	
in	format	The format string containing f's.	
in	values	The floating point values to insert for f. The number of elements in the array must match the number of f's in the format string.	

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintf, snprintfFloat

Gets the length of a 0-terminated unicode string.

Parameters

str	The string in question.
-----	-------------------------

Returns

Length of string.

Gets the length of a 0-terminated string.

Parameters

str The	string.
---------	---------

Returns

Length of string.

7.226.2.12 strncmp()

Compares up to maxchars characters of the string str1 to those of the string str2. This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ, until a terminating null-character is reached, or until maxchars characters match in both strings, whichever happens first.

Parameters

str1	The first string.
str2	The second string.
maxchars	The maximum number of chars to compare.

Returns

Returns an integral value indicating the relationship between the strings: A zero value indicates that the characters compared in both strings are all equal. A value greater than zero indicates that the first character that does not match has a greater value in str1 than in str2; And a value less than zero indicates the opposite.

7.226.2.13 strncmp_ignore_white_spaces()

Like strncmp except that ignore any spaces in the two strings.

Parameters

str1	The first string.
str2	The second string.
maxchars	The maximum number of chars to compare.

Returns

Returns an integral value indicating the relationship between the strings: A zero value indicates that the characters compared in both strings are all equal. A value greater than zero indicates that the first character that does not match has a greater value in str1 than in str2; And a value less than zero indicates the opposite.

uint16_t maxchars) [static]

Copy a string to a destination buffer, UnicodeChar to UnicodeChar version. Stops if it encounters a zero-termination, in which case the zero-termination is included in the destination string. Otherwise copies maxchars.

Parameters

out	dst	The destination buffer. Must have a size of at least maxchars.
in	src	The source string (UnicodeChars)
	maxchars	Maximum number of characters to copy.

Returns

The number of characters copied (excluding zero-termination if encountered)

Warning

If there is no null-termination among the first n UnicodeChars of src, the string placed in destination will NOT be zero-terminated!

```
const char *RESTRICT src,
uint16_t maxchars ) [static]
```

Copy a string to a destination buffer, char to UnicodeChar version. Stops if it encounters a zero-termination, in which case the zero-termination is included in the destination string. Otherwise copies maxchars.

Parameters

out	dst	The destination buffer. Must have a size of at least maxchars.
in	src	The source string as an array of chars.
	maxchars	Maximum number of characters to copy.

Returns

The number of characters copied (excluding zero-termination if encountered)

Warning

If there is no null-termination among the first n bytes of src, the string placed in destination will NOT be zero-terminated!

7.226.2.16 toUTF8()

Converts a string from unicode to utf8. The conversion stops if there is no more room in the destination or if the terminating zero character has been converted. U+10000 through U+10FFFF are skipped.

Parameters

	unicode	The unicode string.
out	utf8	The destination buffer for the converted string.
	maxbytes	The maximum number of bytes that the utf8 array can hold.

Returns

The number of characters successfully converted from unicode to utf8 including the terminating zero.

7.226.2.17 utoa()

Integer to ASCII conversion.

Parameters

	value	to convert.
out	buffer	to place result in.
	bufferSize	Size of buffer (number of 16-bit values).
	radix	to use (8 for octal, 10 for decimal, 16 for hex)

7.226.2.18 vsnprintf() [1/2]

Variant of snprintf. See snprintf for details.

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string.
	pArg	The values to insert in the format string.

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintf

7.226.2.19 vsnprintf() [2/2]

Variant of snprintf. See snprintf for details.

Parameters

out	dst	Buffer for the formatted string.
	dstSize	Size of the dst buffer measured by number of UnicodeChars the buffer can hold.
in	format	The format string.
	pArg	The values to insert in the format string.

Returns

pointer to the first element in the buffer where the formatted string is placed.

See also

snprintf

7.227 Vector < T, capacity > Class Template Reference

A very simple container class using pre-allocated memory.

```
#include <touchgfx/hal/Types.hpp>
```

Public Member Functions

• Vector ()

Default constructor.

T & operator[] (uint16_t idx)

Index operator.

const T & operator[] (uint16_t idx) const

Const version of the index operator.

void add (T e)

Adds an element to the Vector if the Vector is not full.

• void remove (T e)

Removes an element from the Vector if found in the Vector.

• T removeAt (uint16_t index)

Removes an element at the specified index of the Vector.

T quickRemoveAt (uint16_t index)

Removes an element at the specified index of the Vector.

• void reverse ()

Reverses the ordering of the elements in the Vector.

bool contains (T elem)

Checks if the Vector contains an element.

• uint16_t size () const

Gets the current size of the Vector which is the number of elements contained in the Vector.

• bool isEmpty () const

Query if this object is empty.

• uint16_t maxCapacity () const

Query the maximum capacity of the vector.

• void clear ()

Clears the contents of the container.

7.227.1 Detailed Description

```
template < class T, uint16_t capacity > class touchgfx::Vector < T, capacity >
```

A very simple container class using pre-allocated memory.

Template Parameters

	1	The type of objects this container works on.	
		_ ,,,,,,,,	
car	pacity	The maximum number of objects this container can store.	
	1	Tour	chGFX v4.12.3 (C)opyright STMicroelectronics 2014-2019

7.227.2 Constructor & Destructor Documentation

7.227.2.1 Vector()

```
Vector ( ) [inline]
```

Default constructor. Constructs an empty vector.

7.227.3 Member Function Documentation

Adds an element to the Vector if the Vector is not full. Does nothing if the Vector is full.

Parameters

e The element to add to the Vector.

7.227.3.2 clear()

```
void clear ( ) [inline]
```

Clears the contents of the container. It does not destruct any of the elements in the Vector.

7.227.3.3 contains()

```
bool contains (
          T elem ) [inline]
```

Checks if the Vector contains an element. The == operator of the element is used when comparing it with the elements in the Vector.

Parameters

```
elem The element.
```

Returns

true if the Vector contains the element, false otherwise.

7.227.3.4 isEmpty()

```
bool isEmpty ( ) const [inline]
```

Query if this object is empty.

Returns

true if the Vector contains no elements.

7.227.3.5 maxCapacity()

```
uint16_t maxCapacity ( ) const [inline]
```

Query the maximum capacity of the vector.

Returns

The capacity the Vector was initialized with.

```
7.227.3.6 operator[]() [1/2]
```

Index operator.

Parameters

idx The index of the element to obtain.

Returns

A reference to the element placed at index idx.

7.227.3.7 operator[]() [2/2]

Const version of the index operator.

Parameters

```
idx The index of the element to obtain.
```

Returns

A const reference to the element placed at index idx.

7.227.3.8 quickRemoveAt()

Removes an element at the specified index of the Vector. The last element in the list is moved to the position where the element is removed.

Parameters

index	The index to remove.
-------	----------------------

Returns

The value of the removed element.

7.227.3.9 remove()

```
void remove (
          T e ) [inline]
```

Removes an element from the Vector if found in the Vector. Does nothing if the element is not found in the Vector. The == operator of the element is used when comparing it with the elements in the Vector.

Parameters

e The element to remove from the Vector.

7.227.3.10 removeAt()

Removes an element at the specified index of the Vector. Will "bubble-down" any remaining elements after the specified index.

Parameters

!l	The standard to be accessed
inaex	The index to remove.

Returns

The value of the removed element.

7.227.3.11 reverse()

```
void reverse ( ) [inline]
```

Reverses the ordering of the elements in the Vector.

7.227.3.12 size()

```
uint16_t size ( ) const [inline]
```

Gets the current size of the Vector which is the number of elements contained in the Vector.

Returns

The size of the Vector.

7.228 Vector4 Class Reference

This class represents a homogeneous 3D vector.

```
#include <touchgfx/Math3D.hpp>
```

Public Member Functions

• Vector4 ()

Default constructor.

• Vector4 (float x, float y, float z)

Constructor.

Vector4 crossProduct (const Vector4 & operand)

Cross product.

Additional Inherited Members

7.228.1 Detailed Description

This class represents a homogeneous 3D vector.

See also

quadruple

7.228.2 Constructor & Destructor Documentation

```
7.228.2.1 Vector4() [1/2]
Vector4 ( ) [inline]
Default constructor.
```

```
7.228.2.2 Vector4() [2/2]
```

Constructor.

Parameters

Χ	The x value.
У	The y value.
Z	The z value.

7.228.3 Member Function Documentation

7.228.3.1 crossProduct()

Cross product.

Parameters

operand	The second operand.
---------	---------------------

Returns

The result of the operation.

7.229 View < T > Class Template Reference

This is a generic touchgfx::Screen specialization for normal applications.

```
#include <mvp/View.hpp>
```

Public Member Functions

• View ()

Default constructor.

• void bind (T &presenter)

Binds an instance of a specific Presenter type (subclass) to the View instance.

Protected Attributes

• T * presenter

Pointer to the Presenter associated with this view.

Additional Inherited Members

7.229.1 Detailed Description

```
\label{template} \begin{split} \text{template} \! < \! \text{class T} \! > \\ \text{class touchgfx::View} \! < \! \text{T} \! > \end{split}
```

This is a generic touchgfx::Screen specialization for normal applications. It provides a link to the Presenter class.

Note

All views in the application must be a subclass of this type.

Template Parameters

T | The type of Presenter associated with this view.

See also

Screen

7.229.2 Constructor & Destructor Documentation

```
7.229.2.1 View()
View ( ) [inline]
Default constructor.
```

7.229.3 Member Function Documentation

Binds an instance of a specific Presenter type (subclass) to the View instance. This function is called automatically when a new presenter/view pair is activated.

Parameters

7.230 Widget Class Reference

```
A Widget is a Drawable leaf (i.e. not a container).
#include <touchgfx/widgets/Widget.hpp>
```

Public Member Functions

• Widget ()

Default constructor.

virtual ∼Widget ()

Destructor.

virtual void getLastChild (int16_t x, int16_t y, Drawable **last)

Function for obtaining the the last child of this widget that intersects with the specified point.

virtual uint16_t getType () const

For GUI testing only.

Additional Inherited Members

7.230.1 Detailed Description

A Widget is a Drawable leaf (i.e. not a container). It does not currently contain any implementation code, since the Drawable base class handles everything related to leaf nodes. Extend this when implementing custom widgets.

See also

Drawable

7.230.2 Constructor & Destructor Documentation

```
7.230.2.1 Widget()
Widget ( ) [inline]
Default constructor.

7.230.2.2 ~Widget()
~Widget ( ) [inline], [virtual]
Destructor.
```

7.230.3 Member Function Documentation

7.230.3.1 getLastChild()

```
void getLastChild (
    int16_t x,
    int16_t y,
    Drawable ** last ) [inline], [virtual]
```

Function for obtaining the last child of this widget that intersects with the specified point. Used in input event handling for obtaining the appropriate drawable that should receive the event. Note that input events must be delegated to the last drawable of the tree (meaning highest z-order / front-most drawable).

Only containers can have children, so this implementation simply yields itself as result. The container implementation will filter children that do not intersect with the point or are not visible/enabled, so performing those checks are unnecessary.

Parameters

	Х	The point of intersection expressed in coordinates relative to the parent.
	У	The y coordinate.
out	last	Result will be placed here.

Implements Drawable.

7.230.3.2 getType()

```
uint16_t getType ( ) const [inline], [virtual]
```

For GUI testing only. Returns type of this drawable.

Returns

TYPE_WIDGET.

Reimplemented from Drawable.

Reimplemented in TextureMapper, TextArea, TextAreaWithTwoWildcards, RadioButton, AnimatedImage, Scalable—Image, BoxWithBorder, ButtonWithLabel, TiledImage, TextAreaWithOneWildcard, Box, ButtonWithIcon, Image, SnapshotWidget, Button, TouchArea, ToggleButton, and AbstractButton.

7.231 WildcardTextButtonStyle < T > Class Template Reference

A wildcard text button style.

#include <WildcardTextButtonStyle.hpp>

Public Member Functions

• WildcardTextButtonStyle ()

Default constructor.

virtual ~WildcardTextButtonStyle ()

Destructor.

void setWildcardText (TypedText t)

Sets wildcard text.

void setWildcardTextX (int16 t x)

Sets wildcard text x coordinate.

void setWildcardTextY (int16_t y)

Sets wildcard text y coordinate.

void setWildcardTextXY (int16_t x, int16_t y)

Sets wildcard text xy.

• void setWildcardTextPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets text position

void setWildcardTextRotation (TextRotation rotation)

Sets wildcard text rotation.

void setWildcardTextBuffer (const Unicode::UnicodeChar *buffer)

Sets wildcard text buffer.

void setWildcardTextColors (colortype newColorReleased, colortype newColorPressed)

Sets wild card text colors.

Protected Member Functions

virtual void handlePressedUpdated ()

Handles the pressed updated.

· virtual void handleAlphaUpdated ()

Handles the alpha updated.

Protected Attributes

· TextAreaWithOneWildcard wildcardText

The wildcard text.

colortype colorReleased

The color released.

· colortype colorPressed

The color pressed.

7.231.1 Detailed Description

```
\label{template} \mbox{template} < \mbox{class T} > \\ \mbox{class touchgfx::WildcardTextButtonStyle} < \mbox{T} > \\ \mbox{touchgfx::WildcardTextButtonStyle} < \mbox{T} > \\ \mb
```

An wildcard text button style. This class is supposed to be used with one of the ButtonTrigger classes to create a functional button. This class will show a text with a wildcard in one of two colors depending on the state of the button (pressed or released).

The WildcardTextButtonStyle does not set the size of the enclosing container (normally AbstractButtonContainer). The size must be set manually.

To get a background behind the text, use WildcardTextButtonStyle together with e.g. ImageButtonStyle: Wildcard← TextButtonStyle<ImageButtonStyle<ClickButtonTrigger> > myButton;

The position of the text can be adjusted with setTextXY (default is centered).

Template Parameters

T Generic type parameter. Typically a AbstractButtonContainer subclass.

See also

AbstractButtonContainer

Template Parameters

T Generic type parameter.

7.231.2 Member Function Documentation

7.231.2.1 setWildcardText()

Parameters

 $t \mid A \text{ TypedText to process.}$

964 Class Documentation

7.231.2.2 setWildcardTextBuffer()

Parameters

Г	in,out	buffer	If non-null, the buffer.
---	--------	--------	--------------------------

7.231.2.3 setWildcardTextColors()

Parameters

newColorReleased	The new color released.
newColorPressed	The new color pressed.

7.231.2.4 setWildcardTextPosition()

```
void setWildcardTextPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [inline]
```

Parameters

X	The x coordinate.
У	The y coordinate.
width	The width of the text.
height	The height of the text.

7.231.2.5 setWildcardTextRotation()

Parameters

rotation TI	ne rotation.
-------------	--------------

7.231.2.6 setWildcardTextX()

Parameters

```
x The x coordinate.
```

7.231.2.7 setWildcardTextXY()

```
void setWildcardTextXY (
          int16_t x,
          int16_t y) [inline]
```

Parameters

Х	The x coordinate.
У	The y coordinate.

7.231.2.8 setWildcardTextY()

Parameters

y The y coordinate.

7.232 ZoomAnimationImage Class Reference

Class for optimizing and wrapping move and zoom operations on ScalableImages.

```
#include <touchgfx/containers/ZoomAnimationImage.hpp>
```

Public Types

enum ZoomMode {

```
\label{eq:fixed_center} \textbf{FIXED\_CENTER} = 0, \ \textbf{FIXED\_LEFT}, \ \textbf{FIXED\_RIGHT}, \ \textbf{FIXED\_TOP}, \ \textbf{FIXED\_BOTTOM}, \ \textbf{FIXED\_RIGHT\_AND\_BOTTOM}, \ \textbf{FIXED\_RIGHT\_AND\_BOTTOM} \ \}
```

A ZoomMode describes in which direction the image will grow/shrink when do a zoom animation. A FIXED direction means that the image will not grow/shrink in that direction.

Public Member Functions

· ZoomAnimationImage ()

Default constructor.

966 Class Documentation

virtual ~ZoomAnimationImage ()

Destructor.

void startZoomAnimation (int16_t endWidth, int16_t endHeight, uint16_t duration, ZoomMode zoom
 Mode=FIXED_LEFT_AND_TOP, touchgfx::EasingEquation widthProgressionEquation=&touchgfx::EasingEquations::linearEaseNone, EasingEquation heightProgressionEquation=&touchgfx::EasingEquations
 ::linearEaseNone)

Setup and starts the zoom animation.

void startZoomAndMoveAnimation (int16_t endX, int16_t endY, int16_t endWidth, int16_t end↔
 Height, uint16_t duration, ZoomMode zoomMode=FIXED_LEFT_AND_TOP, EasingEquation x↔
 ProgressionEquation=&touchgfx::EasingEquations::linearEaseNone, EasingEquation widthProgressionEquation=&touchgfx↔
 ::EasingEquations::linearEaseNone, EasingEquation heightProgressionEquation=&touchgfx::Easing←
 Equations::linearEaseNone)

Setup and starts the zoom and move animation.

void cancelZoomAnimation ()

Cancel zoom animation.

virtual void handleTickEvent ()

The tick handler.

void setBitmaps (const Bitmap &smallBitmap, const Bitmap &largeBitmap)

Initializes the bitmap of the image to be used.

Bitmap getSmallBitmap () const

Gets the small bitmap.

• Bitmap getLargeBitmap () const

Gets the large bitmap.

• virtual void setPosition (int16_t x, int16_t y, int16_t width, int16_t height)

Sets the size and position of the image, relative to its parent.

virtual void setWidth (int16 t width)

Sets the width of the image.

virtual void setHeight (int16_t height)

Sets the height of the image.

virtual void setDimension (int16 t width, int16 t height)

Sets the width and height of the image.

virtual void setScalingMode (ScalableImage::ScalingAlgorithm mode)

Sets the scaling algorithm of the ScalableImage.

virtual ScalableImage::ScalingAlgorithm getScalingMode ()

Gets the scaling algorithm of the ScalableImage.

virtual void setAlpha (uint8_t alpha)

Sets the alpha channel for the image.

• virtual uint8_t getAlpha () const

Gets the current alpha value.

virtual void setAnimationDelay (uint16_t delay)

Sets a delay on animations done by the ZoomAnimationImage.

• virtual uint16 t getAnimationDelay () const

Gets the current animation delay.

void setAnimationEndedCallback (touchgfx::GenericCallback < const ZoomAnimationImage & > &callback)

Associates an action to be performed when the animation ends.

· virtual bool isRunning () const

Is there currently an animation running.

virtual bool isZoomAnimationRunning () const

Is there currently an animation running.

virtual uint16_t getType () const

For GUI testing only.

Protected Types

enum States { ANIMATE_ZOOM, ANIMATE_ZOOM_AND_MOVE, NO_ANIMATION }

Animation states.

Protected Member Functions

virtual void updateRenderingMethod ()

Chooses the optimal rendering of the image given the current width and height.

virtual void setCurrentState (States state)

Sets the current animation state.

• void startTimerAndSetParameters (int16_t endWidth, int16_t endHeight, uint16_t duration, ZoomMode zoomMode, EasingEquation widthProgressionEquation, EasingEquation heightProgressionEquation)

Starts timer and set parameters.

virtual void updateZoomAnimationDeltaXY ()

Calculates the change in X and Y caused by the zoom animation given the current ZoomMode.

Protected Attributes

· States currentState

The current animation state.

· uint32 t animationCounter

The progress counter for the animation.

uint16_t zoomAnimationDelay

A delay that is applied before animation start. Expressed in ticks.

touchgfx::Bitmap smallBmp

The bitmap representing the small image.

· touchgfx::Bitmap largeBmp

The bitmap representing the large image.

touchgfx::Image image

The image for displaying the bitmap when the width/height is equal one of the bitmaps.

ScalableImage scalableImage

The scalable image for displaying the bitmap when the width/height is not equal one of the bitmaps.

ZoomMode currentZoomMode

The ZoomMode to use by the animation.

· int16 t zoomAnimationStartWidth

Width of the zoom animation start.

int16_t zoomAnimationStartHeight

Height of the zoom animation start.

int16_t zoomAnimationEndWidth

Width of the zoom animation end.int16 t zoomAnimationEndHeight

Height of the zoom animation end.

• int16 t zoomAnimationStartX

The zoom animation start x coordinate.

· int16_t zoomAnimationStartY

The zoom animation start y coordinate.

int16_t zoomAnimationDeltaX

The zoom animation delta x.

int16_t zoomAnimationDeltaY

The zoom animation delta y.

968 Class Documentation

int16_t moveAnimationEndX

The move animation end x coordinate.

int16 t moveAnimationEndY

The move animation end y coordinate.

• uint16_t animationDuration

Duration of the animation.

• EasingEquation zoomAnimationWidthEquation

The zoom animation width equation.

• EasingEquation zoomAnimationHeightEquation

The zoom animation height equation.

EasingEquation moveAnimationXEquation

The move animation x coordinate equation.

• EasingEquation moveAnimationYEquation

The move animation y coordinate equation.

touchgfx::GenericCallback< const ZoomAnimationImage & > * animationEndedAction

The animation ended action.

Additional Inherited Members

7.232.1 Detailed Description

Class for optimizing and wrapping move and zoom operations on ScalableImages. The ZoomAnimationImage takes two bitmaps representing the same image but at a small and a large resolution. These bitmaps should be the sizes that are used when not animating the image. The ZoomAnimationImage will use an Image for displaying the bitmap when its width and height matches one of them. When it does not it will use a ScalableImage instead. The main idea is that the supplied bitmaps should be the end points of the zoom animation so that it ends up using an Image when not animating. This is, however, not a required. You can animate from and to sizes that are not equal the sizes of the bitmaps. The result is a container that has the high performance of an ordinary image when the size matches the prerendered bitmaps. Moreover it supplies easy to use animation functions that lets you zoom and move the image.

Note

Note that since this container uses the ScalableImage it has the same restrictions. That means no 1 bit per pixel mode.

See also

ScalableImage

7.232.2 Member Enumeration Documentation

7.232.2.1 States

enum States [protected]

Enumerator

ANIMATE_ZOOM	Zoom animation state.
ANIMATE_ZOOM_AND_MOVE	Zoom and move animation state.
NO_ANIMATION	No animation state.

7.232.3 Constructor & Destructor Documentation

7.232.3.1 ZoomAnimationImage()

```
ZoomAnimationImage ( )
```

Creates and initialize the ZoomAnimationImage.

7.232.3.2 ~ZoomAnimationImage()

```
\simZoomAnimationImage ( ) [virtual]
```

Destroys the ZoomAnimationImage.

7.232.4 Member Function Documentation

```
7.232.4.1 getAlpha()
```

```
uint8_t getAlpha ( ) const [virtual]
```

Gets the current alpha value.

Returns

The current alpha value.

7.232.4.2 getAnimationDelay()

```
uint16_t getAnimationDelay ( ) const [virtual]
```

Gets the current animation delay.

Returns

The current animation delay. Expressed in ticks.

7.232.4.3 getLargeBitmap()

```
Bitmap getLargeBitmap ( ) const [inline]
```

Gets the large bitmap.

Returns

the large bitmap.

970 Class Documentation

```
7.232.4.4 getScalingMode()
ScalableImage::ScalingAlgorithm getScalingMode ( ) [virtual]
Gets the scaling algorithm of the ScalableImage.
Returns
     the scaling algorithm used.
7.232.4.5 getSmallBitmap()
Bitmap getSmallBitmap ( ) const [inline]
Gets the small bitmap.
Returns
     the small bitmap.
7.232.4.6 getType()
uint16_t getType ( ) const [inline], [virtual]
For GUI testing only. Returns type of this drawable.
Returns
     TYPE_ZOOMANIMATIONIMAGE.
Reimplemented from Container.
7.232.4.7 handleTickEvent()
void handleTickEvent ( ) [virtual]
The tick handler.
Reimplemented from Drawable.
7.232.4.8 isRunning()
bool isRunning ( ) const [virtual]
Is there currently an animation running.
```

Returns

true if there is an animation running.

7.232.4.9 isZoomAnimationRunning()

```
bool isZoomAnimationRunning ( ) const [virtual]
```

Is there currently an animation running.

Returns

true if there is an animation running.

7.232.4.10 setAlpha()

Sets the alpha channel for the image.

Parameters

```
alpha The alpha value. 255 = completely solid.
```

7.232.4.11 setAnimationDelay()

```
void setAnimationDelay ( \label{eq:condition} \mbox{uint16\_t} \ \ delay \ ) \quad \mbox{[virtual]}
```

Sets a delay on animations done by the ZoomAnimationImage. Defaults to 0.

Parameters

delay The delay in ticks

7.232.4.12 setAnimationEndedCallback()

Associates an action to be performed when the animation ends.

Parameters

	callback	The callback to be executed.	The callback will be given a reference to the ZoomAnimationImage.	7
--	----------	------------------------------	---	---

See also

GenericCallback

972 Class Documentation

7.232.4.13 setBitmaps()

Initializes the bitmap of the image to be used. The bitmaps should represent the same image in the two needed static resolutions. Note that it is possible to scale the image beyond the sizes of these bitmaps.

Parameters

smallBitmap	The image in the smallest resolution.
largeBitmap	The image in the largest resolution.

7.232.4.14 setCurrentState()

Sets the current animation state.

Parameters

state T	he new state.
---------	---------------

7.232.4.15 setDimension()

```
void setDimension (
                int16_t width,
                int16_t height ) [virtual]
```

Sets the width and height of the image. Chooses the optimal rendering method afterwards The image is automatically invalidated.

Parameters

width	The new width.
height	The new height.

7.232.4.16 setHeight()

```
void setHeight (
                int16_t height ) [virtual]
```

Sets the height of the image. Chooses the optimal rendering method afterwards. The image is automatically invalidated.

Parameters

1 didiliotoro		
height	The new height.	

Reimplemented from Drawable.

7.232.4.17 setPosition()

```
void setPosition (
    int16_t x,
    int16_t y,
    int16_t width,
    int16_t height ) [virtual]
```

Sets the size and position of the image, relative to its parent. Chooses the optimal rendering method afterwards The image is automatically invalidated.

Parameters

X	The x coordinate of this Drawable.
У	The y coordinate of this Drawable.
width	The width of this Drawable.
height	The height of this Drawable.

Reimplemented from Drawable.

7.232.4.18 setScalingMode()

Sets the scaling algorithm of the ScalableImage.

Parameters

mode	The new mode.
------	---------------

7.232.4.19 setWidth()

```
void setWidth (
                int16_t width ) [virtual]
```

Sets the width of the image. Chooses the optimal rendering method afterwards The image is automatically invalidated.

Parameters

width	The new width.

Reimplemented from Drawable.

974 Class Documentation

7.232.4.20 startTimerAndSetParameters()

```
void startTimerAndSetParameters (
    int16_t endWidth,
    int16_t endHeight,
    uint16_t duration,
    ZoomMode zoomMode,
    EasingEquation widthProgressionEquation,
    EasingEquation heightProgressionEquation ) [protected]
```

Starts timer and set parameters. Contains code shared between startZoomAnimation() and startZoomAndMove Animation(). If both delay and duration is zero, the end position and size is applied and the animation is ended.

Parameters

endWidth	The end width.
endHeight	The end height.
duration	The duration.
zoomMode	The zoom mode.
widthProgressionEquation	The width progression equation.
heightProgressionEquation	The height progression equation.

7.232.4.21 startZoomAndMoveAnimation()

```
void startZoomAndMoveAnimation (
    int16_t endX,
    int16_t endY,
    int16_t endWidth,
    int16_t endHeight,
    uint16_t duration,
    ZoomMode zoomMode = FIXED_LEFT_AND_TOP,
    EasingEquation xProgressionEquation = &touchgfx::EasingEquations::linearEaseNone,
    EasingEquation yProgressionEquation = &touchgfx::EasingEquations::linearEaseNone,
    EasingEquation widthProgressionEquation = &touchgfx::EasingEquations::linearEase←
None,
    EasingEquation heightProgressionEquation = &touchgfx::EasingEquations::linear←
EaseNone )
```

Setup and starts the zoom and move animation. At end of the animation the image will have been resized to the endWidth and endHeight and have moved from its original position to the endX and endY. Please note that the ZoomMode might influence the actual end position since the zoom transformation might change the X and Y of the image. The ZoomMode FIXED_LEFT_AND_TOP ensures that the endX and endY will be the actual end position.

The development of the width, height, X and Y during the animation is described by the supplied EasingEquations. The container is registered as a TimerWidget. Unregistering is handled automatically when the animation has finished.

Parameters

endX	The X position of the image at animation end. Relative to the container or view that holds the ZoomAnimationImage.
endY	The Y position of the image at animation end. Relative to the container or view that holds the ZoomAnimationImage.
endWidth	The width of the image at animation end.
endHeight	The height of the image at animation end.
duration	The duration of the animation measured in ticks.

Parameters

zoomMode	The zoom mode that will be used during the animation. Default = FIXED_LEFT_AND_TOP.
xProgressionEquation	The equation that describes the development of the X position during the animation. Default = EasingEquations::linearEaseNone.
yProgressionEquation	The equation that describes the development of the Y position during the animation. Default = EasingEquations::linearEaseNone.
widthProgressionEquation	The equation that describes the development of the width during the animation. Default = EasingEquations::linearEaseNone.
heightProgressionEquation	The equation that describes the development of the height during the animation. Default = EasingEquations::linearEaseNone.

7.232.4.22 startZoomAnimation()

Setup and starts the zoom animation. At end of the animation the image will have been resized to the end—Width and endHeight. The development of the width and height during the animation is described by the supplied EasingEquations. The container is registered as a TimerWidget. Unregistering is handled automatically when the animation has finished.

Note that the animation follows the specified ZoomMode so the X and Y of the image might change during animation.

Parameters

endWidth	The width of the image at animation end.
endHeight	The height of the image at animation end.
duration	The duration of the animation measured in ticks.
zoomMode	The zoom mode that will be used during the animation. Default = FIXED_LEFT_AND_TOP.
widthProgressionEquation	The equation that describes the development of the width during the animation. Default = EasingEquations::linearEaseNone.
heightProgressionEquation	The equation that describes the development of the height during the animation. Default = EasingEquations::linearEaseNone.

7.232.4.23 updateRenderingMethod()

```
void updateRenderingMethod ( ) [protected], [virtual]
```

Chooses the optimal rendering of the image given the current width and height.

976 Class Documentation

7.232.4.24 updateZoomAnimationDeltaXY()

void updateZoomAnimationDeltaXY () [protected], [virtual]

Calculates the change in X and Y caused by the zoom animation given the current ZoomMode.

Index

\sim AbstractClock	touchgfx::Drawable, 295
touchgfx::AbstractClock, 74	~DrawableList
~AbstractDirectionProgress	touchgfx::DrawableList, 308
touchgfx::AbstractDirectionProgress, 77	~DrawableListItems
~AbstractPainter	touchgfx::DrawableListItems, 316
touchgfx::AbstractPainter, 78	~DrawableListItemsInterface
~AbstractPartition	touchgfx::DrawableListItemsInterface, 318
touchgfx::AbstractPartition, 109	~Event
~AbstractProgressIndicator	touchgfx::Event, 338
touchgfx::AbstractProgressIndicator, 116	~FadeAnimator
~AbstractShape	touchgfx::FadeAnimator, 340
touchgfx::AbstractShape, 122	~FontProvider
~AnimationTextureMapper	touchgfx::FontProvider, 354
touchgfx::AnimationTextureMapper, 147	~GenericCallback
~Box	touchgfx::GenericCallback, 358
touchgfx::Box, 173	touchgfx::GenericCallback< T1, T2, void >, 360
∼BoxProgress	touchgfx::GenericCallback< T1, void, void >, 362
touchgfx::BoxProgress, 176	touchgfx::GenericCallback< void >, 363
~Button	~HAL
touchgfx::Button, 185	touchgfx::HAL, 380
~ButtonController	~I2C
touchgfx::ButtonController, 188	touchgfx::I2C, 412
~CacheableContainer	~I2CTouchController
touchgfx::CacheableContainer, 197	touchgfx::I2CTouchController, 414
~CachedImage	~ImageProgress
touchgfx::CacheableContainer::CachedImage, 200	touchgfx::ImageProgress, 423
~Canvas	~Keyboard
touchgfx::Canvas, 209	touchgfx::Keyboard, 435
~CanvasWidget	~LCD
touchgfx::CanvasWidget, 212	touchgfx::LCD, 443
~CircleProgress	~LineProgress
touchgfx::CircleProgress, 237	touchgfx::LineProgress, 599
~ClickEvent	~ListLayout
touchgfx::ClickEvent, 244	touchgfx::ListLayout, 605
~Container	~MCUInstrumentation
touchgfx::Container, 255	touchgfx::MCUInstrumentation, 617
\sim CoverTransition	~MVPApplication
touchgfx::CoverTransition, 261	touchgfx::MVPApplication, 628
~DMA_Interface	~MVPHeap
touchgfx::DMA_Interface, 281	touchgfx::MVPHeap, 629
~DMA_Queue	~ModalWindow
touchgfx::DMA Queue, 286	touchgfx::ModalWindow, 620
~DigitalClock	~MoveAnimator
touchgfx::DigitalClock, 274	touchgfx::MoveAnimator, 625
~DragEvent	~NoTouchController
touchgfx::DragEvent, 289	touchgfx::NoTouchController, 632
\sim Draggable	~NoTransition
touchgfx::Draggable, 291	touchgfx::NoTransition, 633
~Drawable	~Outline

to colored to Octable a COCT	Aleston at Delinten 77
touchgfx::Outline, 637	AbstractPainter, 77
~Presenter	touchgfx::AbstractPainter, 78
touchgfx::Presenter, 724	AbstractPainterABGR2222, 80
~RadioButton	AbstractPainterARGB2222, 83
touchgfx::RadioButton, 740	AbstractPainterARGB8888, 86
~RadioButtonGroup	AbstractPainterBGRA2222, 89
touchgfx::RadioButtonGroup, 745	AbstractPainterBW, 92
~RenderingBuffer	AbstractPainterGRAY2, 94
touchgfx::RenderingBuffer, 761	AbstractPainterGRAY4, 96
~ScalableImage	AbstractPainterRGB565, 99
touchgfx::ScalableImage, 769	AbstractPainterRGB888, 102
~Scanline	AbstractPainterRGBA2222, 104
touchgfx::Scanline, 774	AbstractPartition, 108
~Screen	touchgfx::AbstractPartition, 109
touchgfx::Screen, 778	AbstractProgressIndicator, 114
~ScrollBase	touchgfx::AbstractProgressIndicator, 115
touchgfx::ScrollBase, 797	AbstractShape, 120
~ScrollList	touchgfx::AbstractShape, 121
touchgfx::ScrollList, 810	AbstractShape::ShapePoint< T >, 833
~ScrollWheel	activate
touchgfx::ScrollWheel, 815	touchgfx::Presenter, 725
~ScrollWheelBase	add
touchgfx::ScrollWheelBase, 817	touchgfx::Container, 255
\sim ScrollWheelWithSelectionStyle	touchgfx::ListLayout, 605
touchgfx::ScrollWheelWithSelectionStyle, 822	touchgfx::ModalWindow, 620
~ScrollableContainer	touchgfx::RadioButtonGroup, 745
touchgfx::ScrollableContainer, 786	touchgfx::Screen, 779
\sim Shape	touchgfx::ScrollableContainer, 786
touchgfx::Shape, 831	touchgfx::SlideMenu, 839
\sim SlideTransition	touchgfx::SwipeContainer, 866
touchgfx::SlideTransition, 857	touchgfx::Vector, 955
\sim Snapper	addCell
touchgfx::Snapper, 859	touchgfx::Scanline, 774
\sim SnapshotWidget	addCover
touchgfx::SnapshotWidget, 862	touchgfx::Cell, 221
\sim TextProgress	addSpan
touchgfx::TextProgress, 889	touchgfx::Scanline, 776
\sim TextureMapper	addToQueue
touchgfx::TextureMapper, 901	touchgfx::DMA_Interface, 281
\sim TouchController	addressIsAddressable
touchgfx::TouchController, 929	touchgfx::FlashDataReader, 343
\sim Transition	advance
touchgfx::Transition, 930	
\sim UIEventListener	touchgfx::GlyphNode, 369
touchgfx::UIEventListener, 940	afterTransition
\sim Widget	touchgfx::Screen, 779
touchgfx::Widget, 961	Alignment
\sim ZoomAnimationImage	touchgfx, 54
touchgfx::ZoomAnimationImage, 969	allocate
	touchgfx::AbstractPartition, 109, 110
abs	allocateAt
touchgfx, 58	touchgfx::AbstractPartition, 110
AbstractButton, 69	allocateBlock
touchgfx::AbstractButton, 70	touchgfx::FrameBufferAllocator, 355
AbstractButtonContainer, 71	touchgfx::ManyBlockAllocator, 610
AbstractClock, 73	touchgfx::SingleBlockAllocator, 834
touchgfx::AbstractClock, 74	allowDMATransfers
AbstractDirectionProgress, 76	touchgfx::HAL, 380
touchgfx::AbstractDirectionProgress, 76	allowHorizontalDrag

touchgfx::ScrollBase, 798	bitDepth
allowVerticalDrag	touchgfx::LCD16bpp, 459
touchgfx::ScrollBase, 798	touchgfx::LCD16bppSerialFlash, 473
AnalogClock, 130	touchgfx::LCD1bpp, 484
angle	touchgfx::LCD24bpp, 496
touchgfx::CWRUtil, 264, 265	touchgfx::LCD2bpp, 508
animateToItem	touchgfx::LCD32bpp, 519
touchgfx::ScrollBase, 798	touchgfx::LCD4bpp, 532
animateToPosition	touchgfx::LCD8bpp ABGR2222, 543
touchgfx::ScrollBase, 799	touchgfx::LCD8bpp_ARGB2222, 554
touchgfx::ScrollWheelBase, 817	touchgfx::LCD8bpp_BGRA2222, 565
animateToState	touchgfx::LCD8bpp_RGBA2222, 576
touchgfx::SlideMenu, 839	touchgfx::LCD, 444
AnimatedImage, 138	Bitmap, 157
touchgfx::AnimatedImage, 139, 140	touchgfx::Bitmap, 160
AnimatedImageButtonStyle < T >, 143	Bitmap::BitmapData, 169
animationEnabled	Bitmap::CacheTableEntry, 200
touchgfx::AnalogClock, 132	Bitmap::DynamicBitmapData, 319
animationEndedHandler	BitmapFormat
touchgfx::SlideMenu, 839	touchgfx::Bitmap, 159
AnimationParameter	blitCopy
touchgfx::AnimationTextureMapper, 146	touchgfx::HAL, 380–382
AnimationState	touchgfx::LCD16bpp, 459, 460
touchgfx::AnimationTextureMapper, 147	touchgfx::LCD16bppSerialFlash, 473
touchgfx::ScrollBase, 797	touchgfx::LCD1bpp, 484, 485
AnimationTextureMapper, 145	touchgfx::LCD24bpp, 496
touchgfx::AnimationTextureMapper, 147	touchgfx::LCD2bpp, 508, 509
AnimationTextureMapper::AnimationSetting, 144	touchgfx::LCD32bpp, 519, 520
appSwitchScreen	touchgfx::LCD4bpp, 532, 533
touchgfx::Application, 151	touchgfx::LCD8bpp_ABGR2222, 543, 544
Application, 149	touchgfx::LCD8bpp_ARGB2222, 554
touchgfx::Application, 151	touchgfx::LCD8bpp_BGRA2222, 565
applyTransformation	touchgfx::LCD8bpp RGBA2222, 576
touchgfx::TextureMapper, 901	touchgfx::LCD, 444
arcsine	blitCopyARGB8888
touchgfx::CWRUtil, 265	touchgfx::HAL, 382
area	touchgfx::LCD16bpp, 461
touchgfx::Rect, 755	touchgfx::LCD16bppSerialFlash, 474
at	touchgfx::LCD24bpp, 497
touchgfx::AbstractPartition, 111	touchgfx::LCD8bpp_ABGR2222, 544
atoi	touchgfx::LCD8bpp_ARGB2222, 555
touchgfx::Unicode, 943	touchgfx::LCD8bpp_BGRA2222, 566
attach	touchgfx::LCD8bpp_RGBA2222, 577
touchgfx::RenderingBuffer, 762	blitCopyAlphaPerPixel
backEaseIn	touchgfx::LCD16bpp, 460
touchgfx::EasingEquations, 321	touchgfx::LCD2bpp, 509
backEaseInOut	touchgfx::LCD4bpp, 533
touchgfx::EasingEquations, 321	touchgfx::LCD8bpp ABGR2222, 544
backEaseOut	touchgfx::LCD8bpp_ARGB2222, 555
touchgfx::EasingEquations, 322	touchgfx::LCD8bpp_BGRA2222, 566
backPorchExited	touchgfx::LCD8bpp_RGBA2222, 577
touchgfx::HAL, 380	blitCopyGlyph
beginFrame	touchgfx::HAL, 383
touchgfx::HAL, 380	blitCopyL8
bind	touchgfx::LCD16bpp, 461
touchgfx::View, 960	touchgfx::LCD16bppSerialFlash, 474
bindTransition	touchgfx::LCD24bpp, 497
touchgfx::Screen, 779	touchgfx::LCD32bpp, 520
- · · · · · · · · · · · · · · · · · · ·	

blitCopyL8_ARGB8888	cache
touchgfx::LCD16bpp, 462	touchgfx::Bitmap, 160
touchgfx::LCD16bppSerialFlash, 475	cacheAll
touchgfx::LCD24bpp, 498	touchgfx::Bitmap, 161
touchgfx::LCD32bpp, 521	cacheDrawOperations
blitCopyL8_RGB565	touchgfx::Application, 152
touchgfx::LCD16bpp, 462	cacheGetAddress
touchgfx::LCD16bppSerialFlash, 475	touchgfx::Bitmap, 161
touchgfx::LCD32bpp, 521	cachelsCached
blitCopyL8_RGB888	touchgfx::Bitmap, 161
touchgfx::LCD16bpp, 463	cacheRemoveBitmap
touchgfx::LCD24bpp, 498	touchgfx::Bitmap, 162
touchgfx::LCD32bpp, 522	cacheReplaceBitmap
blitCopyRGB565	touchgfx::Bitmap, 162
touchgfx::LCD32bpp, 522	cacheTextString
blitCopyRGB888	touchgfx::HAL, 385
touchgfx::LCD32bpp, 523	CacheableContainer, 196
blitCopyRLE	touchgfx::CacheableContainer, 197
touchgfx::LCD1bpp, 485	CacheableContainer::CachedImage, 199
blitFill	CachedImage
touchgfx::HAL, 384	touchgfx::CacheableContainer::CachedImage, 200
BlitOp, 170	calculateAlpha
BlitOperations	touchgfx::Rasterizer, 751
touchgfx, 55	calculateTextHeight
blitSetTransparencyKey	touchgfx::TextAreaWithWildcardBase, 884
touchgfx::HALSDL2, 404	Callback
touchgfx::HAL, 385	touchgfx::Callback, 202
blockCopy	touchgfx::Callback< dest_type, T1, T2, void >,
touchgfx::HALSDL2, 404	203, 204
touchgfx::HAL, 385	touchgfx::Callback< dest_type, T1, void, void >,
bottom touchafy::Poot 755	205
touchgfx::Rect, 755 bounceEaseIn	$touchgfx:: Callback < dest_type, \ void, \ void, \ void >,$
touchgfx::EasingEquations, 322	207
bounceEaseInOut	Callback< dest_type, T1, T2, T3 >, 201
touchgfx::EasingEquations, 323	Callback< dest_type, T1, T2, void >, 203
bounceEaseOut	Callback< dest_type, T1, void, void >, 204
touchgfx::EasingEquations, 323	Callback< dest_type, void, void, void >, 206
Box, 171	cancelAnimationTextureMapperAnimation
touchgfx::Box, 172	touchgfx::AnimationTextureMapper, 147
	19.4 A 1 11
BoxProgress, 175	cancelMoveAnimation
BoxProgress, 175 touchgfx::BoxProgress, 176	touchgfx::MoveAnimator, 625
	touchgfx::MoveAnimator, 625 Canvas, 208
touchgfx::BoxProgress, 176	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209
touchgfx::BoxProgress, 176 BoxWithBorder, 178	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle< T >, 182	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle< T >, 182 Button, 184	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithLcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189 bw	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220 childGeometryChanged
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189 bw touchgfx::PainterBW, 669	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220 childGeometryChanged touchgfx::Drawable, 295
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189 bw touchgfx::PainterBW, 669 CWRUtil, 262	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220 childGeometryChanged touchgfx::Drawable, 295 touchgfx::ScrollableContainer, 787
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189 bw touchgfx::PainterBW, 669 CWRUtil, 262 CWRUtil::Q10, 725	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220 childGeometryChanged touchgfx::Drawable, 295 touchgfx::ScrollableContainer, 787 circEaseIn
touchgfx::BoxProgress, 176 BoxWithBorder, 178 touchgfx::BoxWithBorder, 179 BoxWithBorderButtonStyle < T >, 182 Button, 184 touchgfx::Button, 185 ButtonController, 187 ButtonWithIcon, 189 ButtonWithLabel, 193 touchgfx::ButtonWithLabel, 194 Buttons, 189 bw touchgfx::PainterBW, 669 CWRUtil, 262	touchgfx::MoveAnimator, 625 Canvas, 208 touchgfx::Canvas, 209 CanvasWidget, 211 touchgfx::CanvasWidget, 212 CanvasWidgetRenderer, 215 capacity touchgfx::AbstractPartition, 112 touchgfx::Partition, 716 ceil28_4 touchgfx, 58 Cell, 220 childGeometryChanged touchgfx::Drawable, 295 touchgfx::ScrollableContainer, 787

touchgfx::EasingEquations, 324	touchgfx::HAL, 386
circEaseOut	configureLCDInterrupt
touchgfx::EasingEquations, 324	touchgfx::HALSDL2, 404
Circle, 222	configurePartialFrameBuffer
touchgfx::Circle, 224	touchgfx::HAL, 386
CircleProgress, 235	ConstFont, 251
touchgfx::CircleProgress, 237	touchgfx::ConstFont, 252
clear	Container, 254
touchgfx::AbstractPartition, 112	touchgfx::Container, 255
touchgfx::GPIO, 371	contains
touchgfx::Vector, 955	touchgfx::Container, 256
clearAllTimerWidgets	touchgfx::Vector, 955
touchgfx::Application, 152	convertHandValueToAngle
clearCache	touchgfx::AnalogClock, 132
touchgfx::Bitmap, 162	copyData
clearFadeAnimationEndedAction	touchgfx::FlashDataReader, 343
touchgfx::FadeAnimator, 340	copyFBRegionToMemory
clearMoveAnimationEndedAction	touchgfx::HAL, 386, 387
touchgfx::MoveAnimator, 625	copyFrameBufferRegionToMemory
ClickButtonTrigger, 242	touchgfx::LCD16bpp, 463
ClickEvent, 242	touchgfx::LCD16bppSerialFlash, 475
touchgfx::ClickEvent, 243	touchgfx::LCD1bpp, 486
ClickEventType	touchgfx::LCD24bpp, 498
touchgfx::ClickEvent, 243	touchgfx::LCD2bpp, 510
ClickListener	touchgfx::LCD32bpp, 523
touchgfx::ClickListener, 246	touchgfx::LCD4bpp, 533
ClickListener< T >, 246	touchgfx::LCD8bpp_ABGR2222, 545
ClutFormat	touchgfx::LCD8bpp_ARGB2222, 556
touchgfx::Bitmap, 159	touchgfx::LCD8bpp_BGRA2222, 567
clz	touchgfx::LCD8bpp_RGBA2222, 578
touchgfx, 59	touchgfx::LCD, 445
Color, 247	copyRect
colortype, 249	touchgfx::LCD1bpp, 486
touchgfx::colortype, 250	touchgfx::LCD2bpp, 510
compactCache	touchgfx::LCD4bpp, 534
touchgfx::Bitmap, 162	copyScreenshotToClipboard
compatibleFramebuffer	touchgfx::HALSDL2, 404
touchgfx::AbstractPainter, 78	cosine
concatenateXRotation	touchgfx::CWRUtil, 265, 266
touchgfx::Matrix4x4, 612	CoverTransition
concatenateXScale	touchgfx::CoverTransition, 261
touchgfx::Matrix4x4, 612	CoverTransition< templateDirection >, 260
concatenateXTranslation	CoverTransition< templateDirection >::FullSolidRect,
touchgfx::Matrix4x4, 613	357
concatenateYRotation	crossProduct
touchgfx::Matrix4x4, 613	touchgfx::Vector4, 959
concatenateYScale	cubicEaseIn
touchgfx::Matrix4x4, 613	touchgfx::EasingEquations, 325
concatenate YTranslation	cubicEaseInOut
touchgfx::Matrix4x4, 614	touchgfx::EasingEquations, 325
concatenateZRotation	cubicEaseOut
	touchgfx::EasingEquations, 326
touchgfx::Matrix4x4, 614	DMA Interface 270
concatenateZScale	DMA_Interface, 279
touchgfx::Matrix4x4, 614	touchgfx::DMA_Interface, 281
concatenateZTranslation	DMA_Queue, 285
touchgfx::Matrix4x4, 615	touchgfx::DMA_Queue, 286
configureInterrupts	DMAType
touchgfx::HALSDL2, 404	touchgfx, 56

deactivate	touchgfx::Keyboard, 435
touchgfx::Presenter, 725	touchgfx::LCD24DebugPrinter, 506
DebugPrinter, 270	touchgfx::PixelDataWidget, 719
dec	touchgfx::PreRenderable, 723
touchgfx::AbstractPartition, 112	touchgfx::RadioButton, 740
DigitalClock, 272	touchgfx::ScalableImage, 770
touchgfx::DigitalClock, 274	touchgfx::Screen, 779, 780
Direction	touchgfx::SnapshotWidget, 862
touchgfx, 55	touchgfx::TextArea, 872
DirectionType	touchgfx::TextAreaWithOneWildcard, 879
touchgfx::AbstractDirectionProgress, 76	touchgfx::TextAreaWithTwoWildcards, 881
disableAlpha	touchgfx::TextureMapper, 901
touchgfx::DMA_Interface, 281	touchgfx::TiledImage, 914
disableInterrupts	touchgfx::TouchArea, 925
touchgfx::HALSDL2, 405	drawBorder
touchgfx::HAL, 387	touchgfx::LCD, 447
displayLeadingZeroForHourIndicator	drawCanvasWidget
touchgfx::DigitalClock, 274	touchgfx::AbstractShape, 122
DisplayMode	touchgfx::CanvasWidget, 213
touchgfx::DigitalClock, 273	touchgfx::Circle, 224
DisplayOrientation	touchgfx::Line, 589
touchgfx, 56	drawDrawableInDynamicBitmap
DisplayRotation	touchgfx::HAL, 387
	•
touchgfx, 56	drawGlyph
DisplayTransformation, 277	touchgfx::LCD16bpp, 464
dist_sqr	touchgfx::LCD16bppSerialFlash, 476
touchgfx::Point, 720	touchgfx::LCD1bpp, 487
div255	touchgfx::LCD24bpp, 499
touchgfx::LCD, 446	touchgfx::LCD2bpp, 511
div255g	touchgfx::LCD32bpp, 524
touchgfx::LCD, 446	touchgfx::LCD4bpp, 534
div255rb	touchgfx::LCD8bpp_ABGR2222, 545
touchgfx::LCD, 447	touchgfx::LCD8bpp_ARGB2222, 556
doRotate	touchgfx::LCD8bpp_BGRA2222, 567
touchgfx::HALSDL2, 405	touchgfx::LCD8bpp_RGBA2222, 578
doSampleTouch	touchgfx::LCD, 447
touchgfx::HALSDL2, 405	drawHorizontalLine
doScroll	touchgfx::LCD, 448
touchgfx::ScrollableContainer, 787	drawPartialBitmap
DragEvent, 287	touchgfx::LCD16bpp, 464
touchgfx::DragEvent, 288	touchgfx::LCD16bppSerialFlash, 477
Draggable	touchgfx::LCD1bpp, 488
touchgfx::Draggable, 291	touchgfx::LCD24bpp, 500
Draggable < T >, 291	touchgfx::LCD2bpp, 512
draw	touchgfx::LCD32bpp, 524
touchgfx::Application, 152	touchgfx::LCD4bpp, 535
touchgfx::Box, 173	touchgfx::LCD8bpp_ABGR2222, 546
touchgfx::BoxWithBorder, 180	touchgfx::LCD8bpp_ARGB2222, 557
touchgfx::Button, 186	touchgfx::LCD8bpp_BGRA2222, 568
touchgfx::ButtonWithIcon, 190	touchgfx::LCD8bpp_RGBA2222, 579
touchgfx::ButtonWithLabel, 194	touchgfx::LCD, 449
touchgfx::CanvasWidget, 212	drawRect
touchgfx::Container, 256	touchgfx::LCD, 449
touchgfx::CoverTransition::FullSolidRect, 357	drawString
touchgfx::DebugPrinter, 270	touchgfx::LCD, 449
touchgfx::Drawable, 295	drawStringLTR
touchgfx::Image, 419	touchgfx::LCD, 450
touchgfx::JSMOCHelper, 430	drawStringRTL

touchgfx::LCD, 450	touchgfx::AbstractPartition, 112
drawTextureMapScanLine	touchgfx::Partition, 716, 717
touchgfx::LCD16bpp, 465	element_size
touchgfx::LCD16bppSerialFlash, 477	touchgfx::AbstractPartition, 113
touchgfx::LCD1bpp, 488	touchgfx::Partition, 717
touchgfx::LCD24bpp, 500	enableAlpha
touchgfx::LCD2bpp, 512	touchgfx::DMA_Interface, 282
touchgfx::LCD32bpp, 512	enableCachedMode
touchgfx::LCD4bpp, 536	touchgfx::CacheableContainer, 198
touchgfx::LCD8bpp ABGR2222, 547	enableCopyWithTransparentPixels
touchgfx::LCD8bpp ARGB2222, 557	touchgfx::DMA Interface, 282
touchgfx::LCD8bpp_BGRA2222, 568	enableHorizontalScroll
touchgfx::LCD8bpp_RGBA2222, 579	touchgfx::ScrollableContainer, 787
touchgfx::LCD, 451	enableInterrupts
drawTextureMapTriangle	touchgfx::HALSDL2, 405
touchgfx::LCD, 451	touchgfx::HAL, 388
drawToDynamicBitmap	enableLCDControllerInterrupt
touchgfx::Drawable, 296	touchgfx::HALSDL2, 406
drawTriangle	touchgfx::HAL, 388
touchgfx::ScalableImage, 770	enableMCULoadCalculation
touchgfx::TextureMapper, 902	touchgfx::HAL, 388
drawVerticalLine	enableVerticalScroll
touchgfx::LCD, 452	touchgfx::ScrollableContainer, 788
Drawable, 292	endFrame
touchgfx::Drawable, 295	touchgfx::HAL, 388
DrawableList, 306	evaluatePendingScreenTransition
touchgfx::DrawableList, 308	touchgfx::MVPApplication, 628
DrawableListItems	Event, 337
touchgfx::DrawableListItems, 316	EventType
DrawableListItems< TYPE, SIZE >, 315	touchgfx::Event, 338
DrawableListItemsInterface, 317	execute
DrawableType	touchgfx::Callback, 202
touchgfx::Drawable, 295	touchgfx::Callback< dest_type, T1, T2, void >, 204
DrawingSurface, 318	touchgfx::Callback< dest_type, T1, void, void >,
dstFormat	206
touchgfx::BlitOp, 171	touchgfx::Callback< dest_type, void, void, void >
dynamicBitmapAddSolidRect	208
touchgfx::Bitmap, 163	touchgfx::DMA_Interface, 282
dynamicBitmapCreate	touchgfx::GenericCallback, 359
touchgfx::Bitmap, 163	touchgfx::GenericCallback< T1, T2, void >, 360
dynamicBitmapDelete	touchgfx::GenericCallback< T1, void, void >, 362
touchgfx::Bitmap, 164	touchgfx::GenericCallback< void >, 363
dynamicBitmapGetAddress	executeCompleted
touchgfx::Bitmap, 164	touchgfx::DMA_Interface, 282
dynamicBitmapSetSolidRect	expandToFit
touchgfx::Bitmap, 164	touchgfx::Rect, 755
FasingFaustion	expoEaseIn
EasingEquation	touchgfx::EasingEquations, 327
touchgfx, 54	expoEaseInOut
EasingEquations, 319	touchgfx::EasingEquations, 328
Edge, 335	expoEaseOut
touchgfx::Edge, 336 elasticEaseIn	touchgfx::EasingEquations, 328
touchgfx::EasingEquations, 326	FadeAnimator
elasticEaseInOut	touchgfx::FadeAnimator, 340
touchgfx::EasingEquations, 326	FadeAnimator< T >, 339
elasticEaseOut	fillMemory
touchgfx::EasingEquations, 327	touchgfx::LCD1bpp, 489
element	fillRect
	ini toot

touchgfx::LCD16bpp, 466	touchgfx, 61
touchgfx::LCD16bppSerialFlash, 478	FrameRefreshStrategy
touchgfx::LCD1bpp, 489	touchgfx::HAL, 379
touchgfx::LCD24bpp, 501	framebufferFormat
touchgfx::LCD2bpp, 513	touchgfx::LCD16bpp, 466
touchgfx::LCD32bpp, 525	touchgfx::LCD16bppSerialFlash, 478
touchgfx::LCD4bpp, 536	touchgfx::LCD1bpp, 489
touchgfx::LCD8bpp_ABGR2222, 547	touchgfx::LCD24bpp, 501
touchgfx::LCD8bpp_ABGH2222, 558	touchgfx::LCD2bpp, 513
	touchgfx::LCD32bpp, 526
touchgfx::LCD8bpp_BGRA2222, 569	touchgfx::LCD32bpp, 520
touchgfx::LCD8bpp_RGBA2222, 580	
touchgfx::LCD, 452	touchgfx::LCD8bpp_ABGR2222, 548
FillingRule	touchgfx::LCD8bpp_ARGB2222, 558
touchgfx::Rasterizer, 751	touchgfx::LCD8bpp_BGRA2222, 569
finalizeTransition	touchgfx::LCD8bpp_RGBA2222, 580
touchgfx, 59	touchgfx::LCD, 453
find	framebufferStride
touchgfx::AbstractPartition, 113	touchgfx::LCD16bpp, 466
touchgfx::ConstFont, 252	touchgfx::LCD16bppSerialFlash, 478
first	touchgfx::LCD1bpp, 490
touchgfx::DMA_Queue, 286	touchgfx::LCD24bpp, 501
touchgfx::LockFreeDMA_Queue, 608	touchgfx::LCD2bpp, 513
fixed28 4Mul	touchgfx::LCD32bpp, 526
_	touchgfx::LCD4bpp, 537
touchgfx, 59	touchgfx::LCD8bpp_ABGR2222, 548
fixed28_4ToFloat	touchgfx::LCD8bpp_ARGB2222, 559
touchgfx, 60	touchgfx::LCD8bpp_BGRA2222, 570
FlashDataReader, 342	touchgfx::LCD8bpp_RGBA2222, 581
floatToFixed16_16	*
touchgfx, 60	touchgfx::LCD, 453
floatToFixed28_4	freeBlockAfterTransfer
touchgfx, 60	touchgfx::FrameBufferAllocator, 356
floorDivMod	touchgfx::ManyBlockAllocator, 610
touchgfx, 60	touchgfx::SingleBlockAllocator, 835
flush	fromUTF8
touchgfx::DMA_Interface, 282	touchgfx::Unicode, 943
touchgfx::NoDMA, 630	frontPorchEntered
flushDMA	touchgfx::HAL, 389
touchgfx::HAL, 388	
flushFrameBuffer	GPIO_ID
	touchgfx::GPIO, 371
touchgfx::HALSDL2, 406	GPIO, 371
touchgfx::HAL, 389	gcd
Font, 344	touchgfx, 61
touchgfx::Font, 346	GenericCallback< T1, T2, T3 >, 358
FontManager, 353	GenericCallback< T1, T2, void >, 359
FontProvider, 354	GenericCallback< T1, void, void >, 361
forEachChild	GenericCallback< void >, 362
touchgfx::Container, 256	GestureEvent, 364
forceReportAsSolid	touchgfx::GestureEvent, 365
touchgfx::Box, 173	GestureType
forceState	
touchgfx::ToggleButton, 920	touchgfx::GestureEvent, 364
	Gestures, 366
touchgfx::ToggleButtonTrigger, 922	touchgfx::Gestures, 367
FrameBuffer	get
touchgfx, 56	touchgfx::GPIO, 372
FrameBufferAllocator, 355	touchgfx::LED, 585
FrameBufferAllocatorSignalBlockDrawn	getAbsoluteRect
touchgfx, 61	touchgfx::Drawable, 296
FrameBufferAllocatorWaitOnTransfer	getAlignment

touchgfx::TypedText, 937	touchgfx::SlideMenu, 840
getAllocationCount	getAnimationStep
touchgfx::AbstractPartition, 113	touchgfx::AnimationTextureMapper, 147
getAllowed	getAnimationSteps
touchgfx::DMA_Interface, 282	touchgfx::ScrollBase, 799
getAlpha	getAnimationStorage
touchgfx::AbstractButtonContainer, 72	touchgfx::HAL, 389
touchgfx::Box, 174	getArc
touchgfx::BoxProgress, 176	touchgfx::Circle, 224
touchgfx::BoxWithBorder, 180	getArcEnd
touchgfx::Button, 186	touchgfx::Circle, 225
touchgfx::CanvasWidget, 213	getArcStart
touchgfx::CircleProgress, 237	touchgfx::Circle, 225, 226
touchgfx::DigitalClock, 274	getArgv
touchgfx::Image, 419	touchgfx::HALSDL2, 406
touchgfx::ImageProgress, 423	getAuxiliaryLCD
touchgfx::LineProgress, 600	touchgfx::HAL, 389
touchgfx::PainterABGR2222, 640	getBackgroundHeight
touchgfx::PainterABGR2222Bitmap, 643	touchgfx::ModalWindow, 620
touchgfx::PainterARGB2222, 646	getBackgroundWidth
touchgfx::PainterARGB2222Bitmap, 650	touchgfx::ModalWindow, 621
touchgfx::PainterARGB8888, 653	getBackgroundX
touchgfx::PainterARGB8888Bitmap, 656	touchgfx::SlideMenu, 840
touchgfx::PainterARGB8888L8Bitmap, 659	getBackgroundY
touchgfx::PainterBGRA2222, 663	touchgfx::SlideMenu, 840
touchgfx::PainterBGRA2222Bitmap, 666	getBitmap
touchgfx::PainterGRAY2, 674	touchgfx::Image, 419
touchgfx::PainterGRAY2Bitmap, 677	touchgfx::ImageProgress, 424
touchgfx::PainterGRAY4, 680	touchgfx::ScalableImage, 771
touchgfx::PainterGRAY4Bitmap, 684	touchgfx::TextureMapper, 903
touchgfx::PainterRGB565, 687	getBitmapPositionX
touchgfx::PainterRGB565Bitmap, 690	touchgfx::TextureMapper, 903
touchgfx::PainterRGB565L8Bitmap, 694	getBitmapPositionY
touchgfx::PainterRGB888, 697	touchgfx::TextureMapper, 903
touchgfx::PainterRGB888Bitmap, 701	getBitsPerPixel
touchgfx::PainterRGB888L8Bitmap, 705	touchgfx::Font, 346
touchgfx::PainterRGBA2222, 708	getBlitCaps
touchgfx::PainterRGBA2222Bitmap, 711	touchgfx::DMA_Interface, 283
touchgfx::PixelDataWidget, 719	touchgfx::HAL, 390
touchgfx::RadioButton, 740	touchgfx::NoDMA, 630
touchgfx::ScalableImage, 770	getBlockForTransfer
touchgfx::SnapshotWidget, 862	touchgfx::FrameBufferAllocator, 356
touchgfx::TextArea, 872	touchgfx::ManyBlockAllocator, 610
touchgfx::TextProgress, 889	touchgfx::SingleBlockAllocator, 835
touchgfx::TextureMapper, 902	getBlueColor
touchgfx::ZoomAnimationImage, 969	touchgfx::Color, 248
getAlphaData	touchgfx::LCD16bpp, 466
touchgfx::Bitmap, 165	touchgfx::LCD16bppSerialFlash, 479
getAnchorAtZero	touchgfx::LCD1bpp, 490
touchgfx::ImageProgress, 424	touchgfx::LCD24bpp, 502
getAngle	touchgfx::LCD2bpp, 514
touchgfx::AbstractShape, 122	touchgfx::LCD32bpp, 526
getAnimationDelay	touchgfx::LCD4bpp, 537
touchgfx::ZoomAnimationImage, 969	touchgfx::LCD8bpp_ABGR2222, 548
getAnimationDuration	touchgfx::LCD8bpp_ARGB2222, 559
touchgfx::AnalogClock, 133	touchgfx::LCD8bpp_BGRA2222, 570
touchgfx::SlideMenu, 839	touchgfx::LCD8bpp_RGBA2222, 581
getAnimationEasingEquation	touchgfx::LCD, 453

getBlueFromColor	touchgfx::Outline, 637
touchgfx::LCD16bpp, 467	getCenter
touchgfx::LCD16bppSerialFlash, 479	touchgfx::Circle, 226
touchgfx::LCD1bpp, 490	touchgfx::CircleProgress, 237
touchgfx::LCD24bpp, 502	getCharForKey
touchgfx::LCD2bpp, 514	touchgfx::Keyboard, 436
touchgfx::LCD32bpp, 527	getCharWidth
touchgfx::LCD4bpp, 537	touchgfx::Font, 347
touchgfx::LCD8bpp_ABGR2222, 548	getCircular
touchgfx::LCD8bpp_ARGB2222, 559	touchgfx::DrawableList, 308
touchgfx::LCD8bpp_BGRA2222, 570	touchgfx::ScrollBase, 799
touchgfx::LCD8bpp_RGBA2222, 581	getClientFrameBuffer
getBorderColor	touchgfx::HAL, 390
touchgfx::BoxWithBorder, 180	getCollapsedXCoordinate
getBorderSize	touchgfx::SlideMenu, 840
touchgfx::BoxWithBorder, 180	getCollapsedYCoordinate
getBoundingRect	touchgfx::SlideMenu, 840
touchgfx::TextureMapper, 903	getColor
getBuffer	touchgfx::Box, 174
touchgfx::Keyboard, 435	touchgfx::BoxProgress, 177
getBufferPosition	touchgfx::BoxWithBorder, 180
touchgfx::Keyboard, 436	touchgfx::PainterABGR2222, 640
getButtonController	touchgfx::PainterARGB2222, 647
touchgfx::HAL, 390	touchgfx::PainterARGB8888, 653
getCCConsumed	touchgfx::PainterBGRA2222, 663
touchgfx::MCUInstrumentation, 617	touchgfx::PainterBW, 669
getCPUCycles	touchgfx::PainterGRAY2, 674
touchgfx::HAL, 390	touchgfx::PainterGRAY4, 681
touchgfx::MCUInstrumentation, 617	touchgfx::PainterRGB565, 687
getCacheTopAddress	touchgfx::PainterRGB888, 697
touchgfx::Bitmap, 165	touchgfx::PainterRGBA2222, 708
getCachedAbsX	touchgfx::TextArea, 872
touchgfx::Drawable, 296	touchgfx::TextProgress, 889
touchgfx::JSMOCHelper, 430	getColor32
getCachedAbsY	touchgfx::colortype, 250
touchgfx::Drawable, 296	getColorFrom24BitRGB
touchgfx::JSMOCHelper, 431	touchgfx::Color, 248
getCachedVisibleRect	touchgfx::LCD16bpp, 467
touchgfx::Drawable, 296	touchgfx::LCD16bppSerialFlash, 479
touchgfx::JSMOCHelper, 431	touchgfx::LCD1bpp, 490
getCacheX	touchgfx::LCD24bpp, 502
touchgfx::AbstractShape, 123	touchgfx::LCD2bpp, 514
touchgfx::Shape, 831	touchgfx::LCD32bpp, 527
getCacheY	touchgfx::LCD4bpp, 538
touchgfx::AbstractShape, 123	touchgfx::LCD8bpp_ABGR2222, 549
touchgfx::Shape, 831	touchgfx::LCD8bpp_ARGB2222, 560
getCallbackAreaForCoordinates	touchgfx::LCD8bpp_BGRA2222, 571
touchgfx::Keyboard, 436	touchgfx::LCD8bpp_RGBA2222, 582
getCameraDistance	touchgfx::LCD, 454
touchgfx::TextureMapper, 903	getColorFromRGB
getCameraX	touchgfx::LCD16bpp, 467
touchgfx::TextureMapper, 904	touchgfx::LCD16bppSerialFlash, 480
getCameraY	touchgfx::LCD1bpp, 491
touchgfx::TextureMapper, 904	touchgfx::LCD24bpp, 503
getCapPrecision	touchgfx::LCD2bpp, 515
touchgfx::Circle, 226	touchgfx::LCD32bpp, 527
touchgfx::CircleProgress, 237	touchgfx::LCD4bpp, 538
getCells	touchgfx::LCD8bpp_ABGR2222, 549

touchafyed CD9hnn ADCD2222 E60	touchafy::UAL 201
touchgfx::LCD8bpp_ARGB2222, 560	touchgfx::HAL, 391
touchgfx::LCD8bpp_BGRA2222, 571	getDragAcceleration
touchgfx::LCD8bpp_RGBA2222, 582	touchgfx::ScrollBase, 799
getContainedArea	getDrawable
touchgfx::Container, 256	touchgfx::DrawableListItems, 316
touchgfx::ScrollableContainer, 788	touchgfx::DrawableListItemsInterface, 318
getCornerX	getDrawableIndex
touchgfx::AbstractShape, 123	touchgfx::DrawableList, 308
touchgfx::Shape, 832	getDrawableIndices
getCornerY	touchgfx::DrawableList, 309
touchgfx::AbstractShape, 124	getDrawableMargin
touchgfx::Shape, 832	touchgfx::DrawableList, 309
getCovers	touchgfx::ScrollBase, 800
touchgfx::Scanline::iterator, 429	getDrawableSize
getCurrentHour	touchgfx::DrawableList, 309
touchgfx::AbstractClock, 74	touchgfx::ScrollBase, 800
getCurrentMinute	getElapsedUS
touchgfx::AbstractClock, 74	touchgfx::MCUInstrumentation, 617
getCurrentScreen	getElement
touchgfx::Application, 152	touchgfx::Matrix4x4, 615
getCurrentSecond	touchgfx::Quadruple, 736
touchgfx::AbstractClock, 74	getEllipsisChar
getCurrentlyDisplayedBitmap	touchgfx::Font, 347
touchgfx::Button, 186	getEnd
touchgfx::ImageButtonStyle, 421	touchgfx::Line, 589
touchgfx::RadioButton, 741	touchgfx::LineProgress, 600
getCurrentlyDisplayedIcon	getEndAngle
touchgfx::ButtonWithIcon, 190	touchgfx::CircleProgress, 238
touchgfx::lconButtonStyle, 416	getEventType
getDMAType	touchgfx::ClickEvent, 244
touchgfx::DMA_Interface, 283	touchgfx::DragEvent, 289
touchgfx::HAL, 391	touchgfx::Event, 338
getData	touchgfx::GestureEvent, 365
touchgfx::Bitmap, 165	getExpandDirection
getDataFormatA4	touchgfx::SlideMenu, 840
touchgfx::Font, 347	getExpandedStateTimeout
getDebugPrinter	touchgfx::SlideMenu, 841
touchgfx::Application, 153	getExpandedStateTimer
getDelay	touchgfx::SlideMenu, 841
touchgfx::RepeatButton, 764	getExpandedXCoordinate
touchgfx::RepeatButtonTrigger, 767	touchgfx::SlideMenu, 841
getDeltaX	getExpandedYCoordinate
	touchgfx::SlideMenu, 841
touchgfx::DragEvent, 289	•
getDeltaY	getExtraData
touchgfx::DragEvent, 289	touchgfx::Bitmap, 166
getDeselectionEnabled	getFadeAnimationDelay
touchgfx::RadioButton, 741	touchgfx::FadeAnimator, 340
touchgfx::RadioButtonGroup, 745	getFallbackChar
getDirection	touchgfx::Font, 347
touchgfx::AbstractDirectionProgress, 77	getFingerSize
touchgfx::ListLayout, 605	touchgfx::HAL, 391
getDisplayHeight	getFirstChild
touchgfx::HAL, 390	touchgfx::Container, 257
getDisplayMode	getFont
touchgfx::DigitalClock, 274	touchgfx::FontManager, 353
getDisplayOrientation	touchgfx::FontProvider, 354
touchgfx::HAL, 391	touchgfx::TypedText, 937
getDisplayWidth	getFontHeight

touchgfx::Font, 347	touchgfx::JSMOCHelper, 431
getFontId	touchgfx::RenderingBuffer, 762
touchgfx::TypedText, 937	getHiddenPixelsWhenExpanded
getForce	touchgfx::SlideMenu, 841
touchgfx::ClickEvent, 244	getHorizontal
getFormat	touchgfx::DrawableList, 310
touchgfx::Bitmap, 166	touchgfx::ScrollBase, 800
touchgfx::Bitmap::BitmapData, 170	getHourHandMinuteCorrection
getFrameBufferAllocator	touchgfx::AnalogClock, 133
touchgfx::HAL, 391	getlconX
getFrameRefreshStrategy	touchgfx::ButtonWithIcon, 191
touchgfx::HAL, 392	touchgfx::IconButtonStyle, 416
getFramebufferStride	getIconY
touchgfx::LCD16bpp, 468	touchgfx::ButtonWithIcon, 191
touchgfx::LCD16bppSerialFlash, 480	touchgfx::IconButtonStyle, 416
touchgfx::LCD1bpp, 491	getld
touchgfx::LCD24bpp, 503	touchgfx::Bitmap, 166
touchgfx::LCD2bpp, 515	touchgfx::TypedText, 937
touchgfx::LCD32bpp, 528	getIndentation
touchgfx::LCD4bpp, 539	touchgfx::TextArea, 872
touchgfx::LCD8bpp_ABGR2222, 549	getIndicatorMax
touchgfx::LCD8bpp_ARGB2222, 560	touchgfx::Slider, 848
touchgfx::LCD8bpp_BGRA2222, 571	getIndicatorMin
touchgfx::LCD8bpp_RGBA2222, 582	touchgfx::Slider, 848
getGSUBTable	getIndicatorPositionRangeSize
touchgfx::Font, 349	touchgfx::Slider, 849
getGlyph	getIndicatorRadius
touchgfx::ConstFont, 252	touchgfx::Slider, 849
touchgfx::Font, 348	getInstance
getGreenColor	touchgfx::Application, 153
touchgfx::Color, 248	touchgfx::HAL, 392
touchgfx::LCD16bpp, 468	getInterval
touchgfx::LCD16bppSerialFlash, 480	touchgfx::RepeatButton, 765
touchgfx::LCD1bpp, 491	touchgfx::RepeatButtonTrigger, 767
touchgfx::LCD24bpp, 503	getItem
touchgfx::LCD2bpp, 515	touchgfx::ScrollList, 811
touchgfx::LCD32bpp, 528	getItemIndex
touchgfx::LCD4bpp, 539	touchgfx::DrawableList, 310
touchgfx::LCD8bpp_ABGR2222, 550	getItemSize
touchgfx::LCD8bpp_ARGB2222, 561	touchgfx::DrawableList, 310
touchgfx::LCD8bpp_BGRA2222, 572	getKerning
touchgfx::LCD8bpp_RGBA2222, 583	touchgfx::ConstFont, 253
touchgfx::LCD, 454	touchgfx::Font, 349
getGreenFromColor	touchgfx::InternalFlashFont, 427
touchgfx::LCD16bpp, 468	getKeyForCoordinates
touchgfx::LCD16bppSerialFlash, 481	touchgfx::Keyboard, 436
touchgfx::LCD1bpp, 492	getKeyMappingList
touchgfx::LCD24bpp, 504	touchgfx::Keyboard, 437
touchgfx::LCD2bpp, 516	getLCDRefreshCount
touchgfx::LCD32bpp, 528	touchgfx::HAL, 392
touchgfx::LCD4bpp, 539	getLabelRotation
touchgfx::LCD8bpp_ABGR2222, 550	touchgfx::ButtonWithLabel, 194
touchgfx::LCD8bpp_ARGB2222, 561	getLabelText
touchgfx::LCD8bpp_BGRA2222, 572	touchgfx::ButtonWithLabel, 194
touchgfx::LCD8bpp_RGBA2222, 583	getLanguage
getHeight	touchgfx::Texts, 896
touchgfx::Bitmap, 166	getLargeBitmap
touchgfx::Drawable, 297	touchgfx::ZoomAnimationImage, 969
- · · · · · · · · · · · · · · · · · · ·	- '

getLastChild	touchgfx::ScrollBase, 801
touchgfx::Container, 257	getNumCells
touchgfx::Drawable, 297	touchgfx::Outline, 637
touchgfx::ScrollableContainer, 788	getNumLines
touchgfx::Widget, 961	touchgfx::LCD, 454
getLayout	getNumPix
touchgfx::Keyboard, 437	touchgfx::Scanline::iterator, 429
getLineEndingStyle	getNumPoints
touchgfx::Line, 589	touchgfx::AbstractShape, 124
touchgfx::LineProgress, 600	touchgfx::Shape, 832
getLineWidth	getNumSpans
touchgfx::Circle, 227	touchgfx::Scanline, 776
touchgfx::CircleProgress, 238	getNumberOfDecimals
touchgfx::Line, 590	touchgfx::TextProgress, 889
touchgfx::LineProgress, 600	getNumberOfDrawables
getLinespacing	touchgfx::DrawableList, 311
touchgfx::TextArea, 872	touchgfx::DrawableListItems, 317
getMCULoadPct	touchgfx::DrawableListItemsInterface, 318
touchgfx::HAL, 392	getNumberOfItems
getMaxPixelsLeft	touchgfx::DrawableList, 311
touchgfx::Font, 349	touchgfx::ScrollBase, 801
getMaxPixelsRight	getNumberOfLines
touchgfx::Font, 349	touchgfx::Font, 350
getMaxSwipeItems	getNumberOfPages
touchgfx::ScrollBase, 800	touchgfx::SwipeContainer, 866
getMaxTextHeight	getNumberOfRegisteredTimerWidgets
touchgfx::Font, 350	touchgfx::Application, 153
getMaxValue	getOffset
touchgfx::Slider, 849	touchgfx::DrawableList, 311
getMinValue	touchgfx::ScrollBase, 802
touchgfx::Slider, 849	touchgfx::TiledImage, 914
getMinimalRect	getOldX
touchgfx::AbstractShape, 124	touchgfx::DragEvent, 290
touchgfx::CanvasWidget, 213	getOldY
touchgfx::Circle, 227, 228	touchgfx::DragEvent, 290
touchgfx::Line, 590	getOrigin
getMinimumTextHeight	touchgfx::AbstractShape, 124
touchgfx::Font, 350	getOrigoX
getMinuteHandSecondCorrection	touchgfx::TextureMapper, 904
touchgfx::AnalogClock, 133	getOrigoY
getMissingBufferSize	touchgfx::TextureMapper, 904
touchgfx::CanvasWidgetRenderer, 216	getOrigoZ
getMoveAnimationDelay	touchgfx::TextureMapper, 904
touchgfx::MoveAnimator, 625	getOutlineBuffer
getNearestAlignedOffset	touchgfx::CanvasWidgetRenderer, 216
touchgfx::ScrollBase, 801	getOutlineBufferSize
touchgfx::ScrollList, 811	touchgfx::CanvasWidgetRenderer, 217
getNewX	getPaddingAfter
touchgfx::DragEvent, 289	touchgfx::ScrollList, 811
getNewY	getPaddingBefore
touchgfx::DragEvent, 290	touchgfx::ScrollList, 811
getNextChar	getPainter
touchgfx::TextProvider, 893	touchgfx::CanvasWidget, 214
getNextLigature	getParent
touchgfx::TextProvider, 893, 894	touchgfx::Drawable, 298
getNextSibling	getPixelData
touchgfx::Drawable, 297	touchgfx::ConstFont, 253
getNormalizedOffset	touchgfx::InternalFlashFont, 428
950 151111411204011000	todongiamitornam labili ont, 720

getPositionForItem	getRenderingBuffer
touchgfx::ScrollBase, 802	touchgfx::Renderer, 759
touchgfx::ScrollList, 812	getRequiredNumberOfDrawables
touchgfx::ScrollWheelBase, 817	touchgfx::DrawableList, 311
getPrecision	getRootContainer
touchgfx::Circle, 228	touchgfx::Screen, 780
getPressed	getRotation
touchgfx::AbstractButtonContainer, 72	touchgfx::TextArea, 873
getPressedState	getScale
touchgfx::AbstractButton, 70	touchgfx::AbstractShape, 125
getProgress	touchgfx::TextureMapper, 905
touchgfx::AbstractProgressIndicator, 116	getScalingAlgorithm
getProgressIndicatorHeight	touchgfx::ScalableImage, 771
touchgfx::AbstractProgressIndicator, 116	getScalingMode
getProgressIndicatorWidth	touchgfx::ZoomAnimationImage, 969
touchgfx::AbstractProgressIndicator, 116	getScanlineCounts
getProgressIndicatorX	touchgfx::CanvasWidgetRenderer, 217
touchgfx::AbstractProgressIndicator, 117	getScanlineCovers
getProgressIndicatorY	touchgfx::CanvasWidgetRenderer, 217
touchgfx::AbstractProgressIndicator, 117	getScanlineStartIndices
getRadioButton	touchgfx::CanvasWidgetRenderer, 217
touchgfx::RadioButtonGroup, 746	getScanlineWidth
getRadius	touchgfx::CanvasWidgetRenderer, 217
touchgfx::Circle, 228	getScrolledX
touchgfx::CircleProgress, 238	touchgfx::ScrollableContainer, 788
getRange	getScrolledY
touchgfx::AbstractProgressIndicator, 117, 118	touchgfx::ScrollableContainer, 789
getRect	getSelected
touchgfx::Bitmap, 166	touchgfx::RadioButton, 741
touchgfx::Drawable, 298	getSelectedItem
getRedColor	touchgfx::ScrollWheelBase, 817
touchgfx::Color, 249	getSelectedItemExtraSizeAfter
touchgfx::LCD16bpp, 469	touchgfx::ScrollWheelWithSelectionStyle, 822
touchgfx::LCD16bppSerialFlash, 481	getSelectedItemExtraSizeBefore
touchgfx::LCD1bpp, 492	touchgfx::ScrollWheelWithSelectionStyle, 822
touchgfx::LCD24bpp, 504	getSelectedItemMarginAfter
touchgfx::LCD2bpp, 516	touchgfx::ScrollWheelWithSelectionStyle, 822
touchgfx::LCD32bpp, 528	getSelectedItemMarginBefore
touchgfx::LCD4bpp, 539	touchgfx::ScrollWheelWithSelectionStyle, 823
touchgfx::LCD8bpp ABGR2222, 550	getSelectedItemOffset
touchgfx::LCD8bpp_ARGB2222, 561	touchgfx::ScrollWheelBase, 818
touchgfx::LCD8bpp_BGRA2222, 572	getSelectedRadioButton
touchgfx::LCD8bpp_RGBA2222, 583	touchgfx::RadioButtonGroup, 746
touchgfx::LCD, 455	getSelectedRadioButtonIndex
getRedFromColor	touchgfx::RadioButtonGroup, 746
touchgfx::LCD16bpp, 469	getShadeAlpha
touchgfx::LCD16bppSerialFlash, 481	touchgfx::ModalWindow, 621
touchgfx::LCD1bpp, 492	getShadeColor
touchgfx::LCD24bpp, 504	touchgfx::ModalWindow, 621
touchgfx::LCD2bpp, 516	getSmallBitmap
touchgfx::LCD32bpp, 529	touchgfx::ZoomAnimationImage, 970
touchgfx::LCD4bpp, 540	getSnapping
touchgfx::LCD8bpp_ABGR2222, 551	touchgfx::ScrollList, 812
touchgfx::LCD8bpp_ARGB2222, 562	getSolidRect
touchgfx::LCD8bpp_BGRA2222, 573	touchgfx::Bitmap, 167
touchgfx::LCD8bpp_RGBA2222, 584	touchgfx::Box, 174
getRenderingAlgorithm	touchgfx::BoxWithBorder, 180
touchgfx::TextureMapper, 905	touchgfx::Button, 186
	3

touchgfx::ButtonWithLabel, 195	touchgfx::Application, 153
touchgfx::CanvasWidget, 214	getToggleCanceled
touchgfx::Container, 257	touchgfx::ToggleButtonTrigger, 924
touchgfx::CoverTransition::FullSolidRect, 357	getTouchSampleRate
touchgfx::Drawable, 298	touchgfx::HAL, 393
touchgfx::Image, 419	getType
touchgfx::PixelDataWidget, 719	touchgfx::AbstractButton, 70
touchgfx::RadioButton, 741	touchgfx::AnimatedImage, 140
touchgfx::ScalableImage, 771	touchgfx::Box, 174
touchgfx::SnapshotWidget, 863	touchgfx::BoxWithBorder, 181
touchgfx::TextArea, 873	touchgfx::Button, 186
touchgfx::TextureMapper, 905	touchgfx::ButtonWithIcon, 191
touchgfx::TiledImage, 915	touchgfx::ButtonWithLabel, 195
touchgfx::TouchArea, 925	touchgfx::ClickEvent, 244
getSolidRectAbsolute	touchgfx::Container, 258
touchgfx::Drawable, 298	touchgfx::DragEvent, 290
getSpacingAbove	touchgfx::Drawable, 299
touchgfx::Font, 351	touchgfx::GestureEvent, 365
getStart	touchgfx::Image, 419
touchgfx::Line, 591	touchgfx::Keyboard, 437
touchgfx::LineProgress, 601	touchgfx::ListLayout, 605
getStartAngle	touchgfx::RadioButton, 741
touchgfx::CircleProgress, 238	touchgfx::ScalableImage, 771
getState	touchgfx::ScrollableContainer, 789
touchgfx::SlideMenu, 841	touchgfx::Slider, 850
touchgfx::ToggleButton, 920	touchgfx::SnapshotWidget, 863
getStateChangeButtonX	touchgfx::TextArea, 874
touchgfx::SlideMenu, 842	touchgfx::TextAreaWithOneWildcard, 880
getStateChangeButtonY	touchgfx::TextAreaWithTwoWildcards, 882
touchgfx::SlideMenu, 842	touchgfx::TextureMapper, 905
getStringWidth	touchgfx::TiledImage, 915
touchgfx::Font, 351	touchgfx::ToggleButton, 921
getStringWidthLTR	touchgfx::TouchArea, 926
touchgfx::Font, 352	touchgfx::Widget, 961
getStringWidthRTL	touchgfx::ZoomAnimationImage, 970
touchgfx::Font, 352	getTypedText
getSwipeAcceleration	touchgfx::TextArea, 874
touchgfx::ScrollBase, 802	touchgfx::TextProgress, 890
getTFTCurrentLine	getUsedBufferSize
touchgfx::HAL, 392	touchgfx::CanvasWidgetRenderer, 218
getTFTFrameBuffer	getValue
touchgfx::HALSDL2, 407	touchgfx::AbstractProgressIndicator, 118
touchgfx::HAL, 393	touchgfx::Slider, 850
getText	getValueRangeSize
touchgfx::Texts, 896	touchgfx::Slider, 850
touchgfx::TypedText, 937	getVelocity
getTextDirection	touchgfx::GestureEvent, 365
touchgfx::TypedText, 938	getVisiblePixelsWhenCollapsed
getTextHeight	touchgfx::SlideMenu, 842
-	_
touchgfx::TextArea, 873	getVisibleRect
touchgfx::TextAreaWithOneWildcard, 879	touchgfx::Drawable, 299
touchgfx::TextAreaWithTwoWildcards, 882	getWideTextAction
getTextWidth	touchgfx::TextArea, 874
touchgfx::DigitalClock, 275	getWidget
touchgfx::TextArea, 873	touchgfx::JSMOCHelper, 431
touchgfx::TextAreaWithOneWildcard, 879	getWidth
touchgfx::TextAreaWithTwoWildcards, 882	touchgfx::Bitmap, 167
getTimerWidgetCountForDrawable	touchgfx::Drawable, 299

touchgfx::JSMOCHelper, 431	getX
touchgfx::RenderingBuffer, 762	touchgfx::ClickEvent, 245
getWildcard	touchgfx::Drawable, 299
touchgfx::TextAreaWithOneWildcard, 880	touchgfx::GestureEvent, 366
getWildcard1	touchgfx::Quadruple, 737
touchgfx::TextAreaWithTwoWildcards, 882	getY
getWildcard2	touchgfx::ClickEvent, 245
touchgfx::TextAreaWithTwoWildcards, 883	touchgfx::Drawable, 300
getWindowTitle	touchgfx::GestureEvent, 366
touchgfx::HALSDL2, 407	touchgfx::Quadruple, 737
getWriteMemoryUsageReport	touchgfx::Scanline, 776
touchgfx::CanvasWidgetRenderer, 218	getZ
	touchgfx::Quadruple, 737
getX0	giveFrameBufferSemaphore
touchgfx::TextureMapper, 905	touchgfx::OSWrappers, 634
getX1	giveFrameBufferSemaphoreFromISR
touchgfx::TextureMapper, 906	touchgfx::OSWrappers, 634
getX2	GlyphFlags
touchgfx::TextureMapper, 906	,
getX3	touchgfx, 57
touchgfx::TextureMapper, 906	GlyphNode, 368
getXAdjust	Gradient
touchgfx::RenderingBuffer, 762	touchgfx, 57
getXAngle	Gradients, 373
touchgfx::TextureMapper, 906	touchgfx::Gradients, 373
getXBorder	HALSDL2, 402
touchgfx::ScrollableContainer, 789	touchgfx::HALSDL2, 403
getXOffset	HAL, 374
touchgfx::TiledImage, 915	touchgfx::HAL, 379
getXScrollbar	handleClickEvent
touchgfx::ScrollableContainer, 789	touchgfx::AbstractButton, 70
getY0	touchgfx::Application, 154
touchgfx::TextureMapper, 906	touchgfx::ClickButtonTrigger, 242
getY1	touchgfx::ClickListener, 247
touchgfx::TextureMapper, 907	touchgfx::Drawable, 300
getY2	touchgfx::Keyboard, 437
touchgfx::TextureMapper, 907	touchgfx::RadioButton, 742
getY3	touchgfx::RepeatButton, 765
touchgfx::TextureMapper, 907	touchgfx::RepeatButtonTrigger, 767
getYAngle	touchgfx::Screen, 780
touchgfx::TextureMapper, 907	touchgfx::ScrollList, 812
getYBorder	touchgfx::ScrollWheelBase, 818
touchgfx::ScrollableContainer, 790	touchgfx::ScrollableContainer, 790
getYOffset	touchgfx::Slider, 850
touchgfx::TiledImage, 915	touchgfx::Snapper, 859
getYScrollbar	touchgfx::SwipeContainer, 867
touchgfx::ScrollableContainer, 790	touchgfx::ToggleButton, 921
getZ0	touchgfx::ToggleButtonTrigger, 924
touchgfx::TextureMapper, 907	touchgfx::TouchArea, 926
getZ1	touchgfx::TouchButtonTrigger, 927
touchgfx::TextureMapper, 908	touchgfx::UIEventListener, 940
getZ2	handleDragEvent
touchgfx::TextureMapper, 908	touchgfx::Application, 154
getZ3	touchgfx::Draggable, 291
touchgfx::TextureMapper, 908	touchgfx::Drawable, 300
getZAngle	touchgfx::Keyboard, 438
touchgfx::TextureMapper, 908	touchgfx::Screen, 780
getW	touchgfx::ScrollBase, 803
touchgfx::Quadruple, 736	touchgfx::ScrollWheelBase, 818
· '	3

t	tarrale of colOOTarrale Oarstoollan, 44.4
touchgfx::ScrollableContainer, 790	touchgfx::I2CTouchController, 414
touchgfx::Slider, 851	I2C, 411
touchgfx::Snapper, 860	touchgfx::I2C, 412
touchgfx::SwipeContainer, 867	IconButtonStyle< T >, 415
touchgfx::TouchArea, 926	identity
touchgfx::UIEventListener, 940	touchgfx::Matrix4x4, 615
handleGestureEvent	Image, 417
touchgfx::Application, 154	touchgfx::Image, 418
touchgfx::Drawable, 300	ImageButtonStyle< T >, 420
touchgfx::Screen, 781	ImageProgress, 422
touchgfx::ScrollBase, 803	
touchgfx::ScrollWheelBase, 818	touchgfx::ImageProgress, 423
-	inbox
touchgfx::ScrollableContainer, 791	touchgfx::RenderingBuffer, 763
touchgfx::SwipeContainer, 867	includes
touchgfx::UIEventListener, 940	touchgfx::Rect, 756
handleKeyEvent	indexOf
touchgfx::Application, 154	touchgfx::AbstractPartition, 114
touchgfx::Screen, 781	init
touchgfx::UIEventListener, 941	touchgfx::ButtonController, 188
handlePendingScreenTransition	touchgfx::Buttons, 189
touchgfx::Application, 155	touchgfx::CoverTransition, 261
touchgfx::MVPApplication, 628	
touchgfx::UIEventListener, 941	touchgfx::GPIO, 372
handleTickEvent	touchgfx::I2CTouchController, 414
touchgfx::AnimatedImage, 140	touchgfx::I2C, 412
	touchgfx::LCD16bpp, 469
touchgfx::AnimationTextureMapper, 147	touchgfx::LCD16bppSerialFlash, 482
touchgfx::Application, 155	touchgfx::LCD24bpp, 504
touchgfx::CoverTransition, 261	touchgfx::LCD2bpp, 516
touchgfx::Drawable, 301	touchgfx::LCD32bpp, 529
touchgfx::FadeAnimator, 341	touchgfx::LCD4bpp, 540
touchgfx::MoveAnimator, 625	touchgfx::LCD8bpp_ABGR2222, 551
touchgfx::NoTransition, 633	touchgfx::LCD8bpp_ARGB2222, 562
touchgfx::RepeatButton, 765	touchgfx::LCD8bpp_BGRA2222, 573
touchgfx::Screen, 781	
touchgfx::ScrollBase, 803	touchgfx::LCD8bpp_RGBA2222, 584
touchgfx::ScrollableContainer, 791	touchgfx::LCD, 455
touchgfx::SlideTransition, 857	touchgfx::LED, 585
touchgfx::SwipeContainer, 867	touchgfx::MCUInstrumentation, 618
touchgfx::Transition, 931	touchgfx::NoTouchController, 632
touchgfx::UIEventListener, 941	touchgfx::SDL2TouchController, 828
touchgfx::ZoomAnimationImage, 970	touchgfx::SDLTouchController, 829
-	touchgfx::SlideTransition, 857
hasBlockReadyForTransfer	touchgfx::TouchController, 929
touchgfx::FrameBufferAllocator, 356	touchgfx::Transition, 931
touchgfx::ManyBlockAllocator, 611	initMoveDrawable
touchgfx::SingleBlockAllocator, 835	touchgfx::CoverTransition, 261
hasBuffer	touchgfx::SlideTransition, 857
touchgfx::CanvasWidgetRenderer, 218	initialize
hasTransparentPixels	
touchgfx::Bitmap, 167	touchgfx::DMA_Interface, 283
hasValidId	touchgfx::HAL, 393
touchgfx::TypedText, 938	touchgfx::OSWrappers, 634
height	touchgfx::ScrollBase, 803
touchgfx::GlyphNode, 369	touchgfx::ScrollWheelWithSelectionStyle, 823
hide	touchgfx::TextProvider, 895
touchgfx::ModalWindow, 621	initializeTime12Hour
_	touchgfx::AnalogClock, 134
hw_init	initializeTime24Hour
touchgfx, 61	touchgfx::AnalogClock, 134
I2CTouchController, 413	insert
120 TOUCHOUTHOUGH, 410	III JOI (

touchgfx::Container, 258	touchgfx::ScrollableContainer, 791
touchgfx::ListLayout, 605	isShowing
instance	touchgfx::ModalWindow, 621
	isTextureMapperAnimationRunning
touchgfx::Application, 157	touchgfx::AnimationTextureMapper, 148
InternalFlashFont, 426	isTouchable
touchgfx::InternalFlashFont, 427	
intersect	touchgfx::Drawable, 302
touchgfx::Rect, 756	isValid
invalidate	touchgfx::Callback, 202
touchgfx::CanvasWidget, 214	touchgfx::Callback< dest_type, T1, T2, void >, 204
touchgfx::Drawable, 301	touchgfx::Callback< dest_type, T1, void, void >
invalidateArea	206
touchgfx::Application, 155	touchgfx::Callback< dest_type, void, void, void >
invalidateRect	208
touchgfx::CacheableContainer, 198	touchgfx::GenericCallback, 359
touchgfx::Drawable, 301	touchgfx::GenericCallback< T1, T2, void >, 361
invalidateScrollbars	touchgfx::GenericCallback< T1, void, void >, 362
touchgfx::ScrollableContainer, 791	touchgfx::GenericCallback< void >, 363
isAlphaPerPixel	isVisible
touchgfx::Bitmap, 167	touchgfx::Drawable, 302
isAnimatedImageRunning	isZoomAnimationRunning
touchgfx::AnimatedImage, 140	touchgfx::ZoomAnimationImage, 970
isAnimating	itemChanged
touchgfx::ScrollBase, 803	touchgfx::DrawableList, 312
isChildInvalidated	touchgfx::ScrollBase, 804
touchgfx::CacheableContainer, 198	touchgfx::ScrollWheelWithSelectionStyle, 823
	iterator
isDMARunning	touchgfx::Scanline::iterator, 428
touchgfx::DMA_Interface, 284	itoa
isDmaQueueEmpty	touchgfx::Unicode, 943
touchgfx::DMA_Interface, 283	todong/kilonioodo, o to
isDmaQueueFull	JSMOCHelper, 429
touchgfx::DMA_Interface, 283	touchgfx::JSMOCHelper, 430
isDone	JSMOC
touchgfx::Transition, 931	touchgfx::Screen, 781
isEmpty	g,
touchgfx::DMA_Queue, 286	keepOffsetInsideLimits
touchgfx::LockFreeDMA_Queue, 608	touchgfx::ScrollBase, 804
touchgfx::Rect, 756	touchgfx::ScrollList, 813
touchgfx::Vector, 955	touchgfx::ScrollWheelBase, 819
isFadeAnimationRunning	KerningNode, 432
touchgfx::FadeAnimator, 341	kerningTablePos
isFull	touchgfx::GlyphNode, 370
touchgfx::DMA_Queue, 287	Keyboard, 433
touchgfx::LockFreeDMA_Queue, 608	touchgfx::Keyboard, 435
isMoveAnimationRunning	Keyboard::CallbackArea, 208
touchgfx::MoveAnimator, 625	Keyboard::Key, 432
isPreRendered	Keyboard::KeyMapping, 440
touchgfx::PreRenderable, 723	Keyboard::KeyMapping, 440 Keyboard::KeyMappingList, 440
isReady	Keyboard::Layout, 440
touchgfx::Scanline, 776	ReyboardLayout, 440
isReverse	LCD16bpp, 457
touchgfx::AnimatedImage, 141	LCD16bppSerialFlash, 470
-	• •
isRunning touchafy::AnimatedImage, 141	touchgfx::LCD16bppSerialFlash, 472
touchgfx::AnimatedImage, 141	LCD1bpp, 482
touchgfx::FadeAnimator, 341	LCD24DebugPrinter, 505
touchgfx::MoveAnimator, 626	LCD24bpp, 493
touchgfx::ZoomAnimationImage, 970	LCD2bpp, 506
isScrollableXY	LCD2getPixel

touchgfx, 61, 62	touchgfx::MCUInstrumentation, 617
LCD2setPixel	MVPApplication, 627
touchgfx, 62	touchgfx::MVPApplication, 628
LCD2shiftVal	MVPHeap, 628
touchgfx, 63	touchgfx::MVPHeap, 629
LCD32bpp, 517	makeSnapshot
LCD4bpp, 530	touchgfx::SnapshotWidget, 863
LCD4getPixel	makeTransition
touchgfx, 63	touchgfx, 65
LCD4setPixel	ManyBlockAllocator< block_size, blocks, bytes_pr_pixe
touchgfx, 64	>, 609
LCD8bpp ABGR2222, 541	markBlockReadyForTransfer
LCD8bpp_ARGB2222, 552	touchgfx::FrameBufferAllocator, 356
LCD8bpp_BGRA2222, 563	touchgfx::ManyBlockAllocator, 611
LCD8bpp_RGBA2222, 574	touchgfx::SingleBlockAllocator, 836
LCD::StringVisuals, 864	Matrix4x4, 611
LCD, 441	touchgfx::Matrix4x4, 612
LED, 585	maxCapacity
LINE ENDING STYLE	touchgfx::Vector, 956
touchgfx::Line, 588	memset
lcd	touchgfx, 65
touchgfx::HAL, 393	mixColors
Line, 586	touchgfx::AbstractPainterABGR2222, 81
touchgfx::Line, 588	touchgfx::AbstractPainterABGB2222, 84
LineProgress, 598	touchgfx::AbstractPainterBGRA2222, 90
touchgfx::LineProgress, 599	touchgfx::AbstractPainterBGB565, 100
lineTo	touchgfx::AbstractPainterRGBA2222, 105, 106
	ModalWindow, 618
touchgfx::Canvas, 209, 210	
touchgfx::Outline, 637	touchgfx::ModalWindow, 620
touchgfx::Rasterizer, 751	MoveAnimator
linearEaseIn	touchgfx::MoveAnimator, 624
touchgfx::EasingEquations, 329	MoveAnimator < T >, 623
linearEaseInOut	moveChildrenRelative
touchgfx::EasingEquations, 329	touchgfx::Container, 258
linearEaseNone	touchgfx::ScrollableContainer, 791
touchgfx::EasingEquations, 329	moveOrigin
linearEaseOut	touchgfx::AbstractShape, 125
touchgfx::EasingEquations, 330	moveRelative
ListLayout, 604	touchgfx::Drawable, 302
touchgfx::ListLayout, 604	moveTo
loadSkin	touchgfx::Canvas, 210, 211
touchgfx::HALSDL2, 407	touchgfx::Drawable, 302
lockDMAToFrontPorch	touchgfx::Outline, 638
touchgfx::HAL, 394	touchgfx::Rasterizer, 752
lockFrameBuffer	mulQ5
touchgfx::HAL, 394	touchgfx::CWRUtil, 267
LockFreeDMA_Queue, 607	muldiv
touchgfx::LockFreeDMA_Queue, 607	touchgfx, 66
IookupBilinearRenderVariant	muldivQ10
touchgfx, 64	touchgfx::CWRUtil, 266
IookupNearestNeighborRenderVariant	muldivQ5
touchgfx, 64	touchgfx::CWRUtil, 266
lookupRenderVariant	
touchgfx::ScalableImage, 772	next
touchgfx::TextureMapper, 908	touchgfx::Scanline::iterator, 429
,	nextFadeAnimationStep
MAX_TIMER_WIDGETS	touchgfx::FadeAnimator, 341
touchgfx::Application, 157	nextLine
MCUInstrumentation, 616	touchgfx::LCD16bpp, 469

touchgfx::LCD16bppSerialFlash, 482	touchgfx::CWRUtil::Q5, 733
touchgfx::LCD1bpp, 493	operator-
touchgfx::LCD24bpp, 505	touchgfx::CWRUtil::Q10, 727
touchgfx::LCD2bpp, 517	touchgfx::CWRUtil::Q15, 729
touchgfx::LCD32bpp, 529	touchgfx::CWRUtil::Q5, 733
- · · ·	_
touchgfx::LCD4bpp, 540	operator/
touchgfx::LCD8bpp_ABGR2222, 551	touchgfx::CWRUtil::Q10, 727
touchgfx::LCD8bpp_ARGB2222, 562	touchgfx::CWRUtil::Q15, 730
touchgfx::LCD8bpp BGRA2222, 573	touchgfx::CWRUtil::Q5, 734
touchgfx::LCD8bpp_RGBA2222, 584	operator==
nextMoveAnimationStep	touchgfx::Bitmap, 168
touchgfx::MoveAnimator, 626	- ·
	touchgfx::Rect, 758
nextPixel	operator&
touchgfx::LCD16bpp, 470	touchgfx::Rect, 757
touchgfx::LCD16bppSerialFlash, 482	operator&=
touchgfx::LCD1bpp, 493	touchgfx::Rect, 757
touchgfx::LCD24bpp, 505	operator[]
touchgfx::LCD2bpp, 517	touchgfx::DrawableListItems, 317
touchgfx::LCD32bpp, 530	touchgfx::Vector, 956
- · · ·	•
touchgfx::LCD4bpp, 541	Outline, 636
touchgfx::LCD8bpp_ABGR2222, 551	touchgfx::Outline, 637
touchgfx::LCD8bpp_ARGB2222, 562	OutlineFlags_t
touchgfx::LCD8bpp_BGRA2222, 573	touchgfx::Outline, 637
touchgfx::LCD8bpp RGBA2222, 584	
NoDMA, 629	packedCoord
	touchgfx::Cell, 221
touchgfx::NoDMA, 630	PainterABGR2222, 639
noTouch	touchgfx::PainterABGR2222, 640
touchgfx::HAL, 394	
NoTouchController, 631	PainterABGR2222Bitmap, 642
NoTransition, 633	touchgfx::PainterABGR2222Bitmap, 643
touchgfx::NoTransition, 633	PainterARGB2222, 645
numCellsMissing	touchgfx::PainterARGB2222, 646
touchgfx::CanvasWidgetRenderer, 218	PainterARGB2222Bitmap, 648
-	touchgfx::PainterARGB2222Bitmap, 649
numCellsUsed	PainterARGB8888, 652
touchgfx::CanvasWidgetRenderer, 218	touchgfx::PainterARGB8888, 652
0014	
OSWrappers, 634	PainterARGB8888Bitmap, 655
off	touchgfx::PainterARGB8888Bitmap, 656
touchgfx::LED, 586	PainterARGB8888L8Bitmap, 658
on	touchgfx::PainterARGB8888L8Bitmap, 659
touchgfx::LED, 586	PainterBGRA2222, 661
operation	touchgfx::PainterBGRA2222, 662
•	_
touchgfx::BlitOp, 171	PainterBGRA2222Bitmap, 665
operator int	touchgfx::PainterBGRA2222Bitmap, 666
touchgfx::CWRUtil::Q10, 726	PainterBWBitmap, 670
touchgfx::CWRUtil::Q15, 729	touchgfx::PainterBWBitmap, 671
touchgfx::CWRUtil::Q5, 731	PainterBW, 668
operator uint16_t	PainterGRAY2, 673
touchgfx::colortype, 250	touchgfx::PainterGRAY2, 674
	_
operator!=	PainterGRAY2Bitmap, 676
touchgfx::Bitmap, 167	touchgfx::PainterGRAY2Bitmap, 677
touchgfx::Rect, 757	PainterGRAY4, 679
operator*	touchgfx::PainterGRAY4, 680
touchgfx, 66	PainterGRAY4Bitmap, 682
touchgfx::CWRUtil::Q10, 726	touchgfx::PainterGRAY4Bitmap, 683
touchgfx::CWRUtil::Q5, 732	PainterRGB565, 686
operator+	touchgfx::PainterRGB565, 687
touchgfx::CWRUtil::Q10, 727	PainterRGB565Bitmap, 689
touchgfx::CWRUtil::Q15, 729	touchgfx::PainterRGB565Bitmap, 690

PainterRGB565L8Bitmap, 692	touchgfx::EasingEquations, 331
touchgfx::PainterRGB565L8Bitmap, 693	Quadruple, 735
PainterRGB888, 696	touchgfx::Quadruple, 736
touchgfx::PainterRGB888, 697	quartEaseIn
PainterRGB888Bitmap, 700	touchgfx::EasingEquations, 331
touchgfx::PainterRGB888Bitmap, 701	quartEaseInOut
PainterRGB888L8Bitmap, 703	touchgfx::EasingEquations, 332
touchgfx::PainterRGB888L8Bitmap, 704	quartEaseOut
PainterRGBA2222, 707	touchgfx::EasingEquations, 332
touchgfx::PainterRGBA2222, 708	quickRemoveAt
PainterRGBA2222Bitmap, 710	touchgfx::Vector, 956
touchgfx::PainterRGBA2222Bitmap, 711	quintEaseIn
Pair	touchgfx::EasingEquations, 333
touchgfx::Pair, 714	quintEaseInOut
Pair $< T1, T2 >, 713$	touchgfx::EasingEquations, 333
Partition	quintEaseOut
touchgfx::Partition, 716	touchgfx::EasingEquations, 334
Partition < ListOfTypes, NUMBER_OF_ELEMENTS >,	D # D # 700
715	RadioButton, 738
pauseAnimation	touchgfx::RadioButton, 740
touchgfx::AnimatedImage, 141	radioButtonClickedHandler
performDisplayOrientationChange	touchgfx::RadioButtonGroup, 746
touchgfx::HALSDL2, 408	radioButtonDeselectedHandler
touchgfx::HAL, 394	touchgfx::RadioButtonGroup, 747
PixelDataWidget, 718	RadioButtonGroup
touchgfx::PixelDataWidget, 718	touchgfx::RadioButtonGroup, 745
Point, 720	RadioButtonGroup < CAPACITY >, 743
Point3D, 721	Rasterizer, 749
Point4, 721	touchgfx::Rasterizer, 751
touchgfx::Point4, 722	readRegister
рор	touchgfx::I2C, 412
touchgfx::DMA_Queue, 287	realX
touchgfx::LockFreeDMA_Queue, 608	touchgfx::LCD, 455
positionToValue	realY
touchgfx::Slider, 851	touchgfx::LCD, 456
preRender	Rect, 753
touchgfx::PreRenderable, 723	touchgfx::Rect, 754
PreRenderable	refreshDrawableListsLayout
touchgfx::PreRenderable, 723	touchgfx::ScrollWheelWithSelectionStyle, 823
PreRenderable < T >, 722	refreshDrawables
prepareTransition	touchgfx::DrawableList, 312
touchgfx, 67	region
Presenter, 724	touchgfx::DebugPrinter, 271
touchgfx::Presenter, 725	registerBitmapDatabase
pushCopyOf	touchgfx::Bitmap, 168
touchgfx::DMA_Queue, 287	registerClickEvent
touchgfx::LockFreeDMA_Queue, 609	touchgfx::Gestures, 367
	registerDragEvent
Q10	touchgfx::Gestures, 367
touchgfx::CWRUtil::Q10, 726	registerEventListener
Q15	touchgfx::Gestures, 368
touchgfx::CWRUtil::Q15, 729	touchgfx::HAL, 395
Q5	registerTaskDelayFunction
touchgfx::CWRUtil::Q5, 731	touchgfx::HAL, 395
quadEaseIn	registerTextCache
touchgfx::EasingEquations, 330	touchgfx::HAL, 395
quadEaseInOut	registerTexts
touchgfx::EasingEquations, 331	touchgfx::TypedText, 938
quadEaseOut	registerTimerWidget

touchgfx::Application, 155	touchgfx::AbstractPainterARGB2222, 85
registerTypedTextDatabase	touchgfx::AbstractPainterARGB8888, 88
touchgfx::TypedText, 938	touchgfx::AbstractPainterBGRA2222, 91
remove	touchgfx::AbstractPainterBW, 94
touchgfx::Container, 259	touchgfx::AbstractPainterGRAY2, 95
touchgfx::ListLayout, 606	touchgfx::AbstractPainterGRAY4, 98
touchgfx::ModalWindow, 621	touchgfx::AbstractPainterRGB565, 101
touchgfx::Screen, 781	touchgfx::AbstractPainterRGB888, 103
touchgfx::SlideMenu, 842	touchgfx::AbstractPainterRGBA2222, 107
touchgfx::SwipeContainer, 868	touchgfx::PainterABGR2222Bitmap, 644
touchgfx::Vector, 957	touchgfx::PainterARGB2222Bitmap, 650
removeAll	touchgfx::PainterARGB8888Bitmap, 657
touchgfx::Container, 259	touchgfx::PainterARGB8888L8Bitmap, 660
touchgfx::ListLayout, 606	touchgfx::PainterBGRA2222Bitmap, 667
removeAt	touchgfx::PainterBWBitmap, 672
touchgfx::Vector, 957	touchgfx::PainterGRAY2Bitmap, 678
removeCache	touchgfx::PainterGRAY4Bitmap, 685
touchgfx::Bitmap, 168	touchgfx::PainterGHAT4Bitmap, 693
	touchgfx::PainterRGB565L8Bitmap, 694
render	-
touchgfx::AbstractPainter, 79	touchgfx::PainterRGB888Bitmap, 702
touchgfx::AbstractPainterABGR2222, 82	touchgfx::PainterRGB888L8Bitmap, 705
touchgfx::AbstractPainterARGB2222, 85	touchgfx::PainterRGBA2222Bitmap, 712
touchgfx::AbstractPainterARGB8888, 87	renderLCD_FrameBufferToMemory
touchgfx::AbstractPainterBGRA2222, 91	touchgfx::HALSDL2, 408
touchgfx::AbstractPainterBW, 93	renderNext
touchgfx::AbstractPainterGRAY2, 95	touchgfx::AbstractPainterABGR2222, 82
touchgfx::AbstractPainterGRAY4, 97	touchgfx::AbstractPainterARGB2222, 86
touchgfx::AbstractPainterRGB565, 101	touchgfx::AbstractPainterARGB8888, 88
touchgfx::AbstractPainterRGB888, 103	touchgfx::AbstractPainterBGRA2222, 92
touchgfx::AbstractPainterRGBA2222, 106	touchgfx::AbstractPainterBW, 94
touchgfx::Canvas, 211	touchgfx::AbstractPainterGRAY2, 96
touchgfx::PainterABGR2222, 640	touchgfx::AbstractPainterGRAY4, 98
touchgfx::PainterABGR2222Bitmap, 643	touchgfx::AbstractPainterRGB565, 101
touchgfx::PainterARGB2222, 647	touchgfx::AbstractPainterRGB888, 104
touchgfx::PainterARGB2222Bitmap, 650	touchgfx::AbstractPainterRGBA2222, 107
touchgfx::PainterARGB8888, 653	touchgfx::PainterABGR2222, 641
touchgfx::PainterARGB8888Bitmap, 656	touchgfx::PainterABGR2222Bitmap, 644
touchgfx::PainterARGB8888L8Bitmap, 660	touchgfx::PainterARGB2222, 647
touchgfx::PainterBGRA2222, 663	touchgfx::PainterARGB2222Bitmap, 651
touchgfx::PainterBGRA2222Bitmap, 666	touchgfx::PainterARGB8888, 654
touchgfx::PainterBWBitmap, 672	touchgfx::PainterARGB8888Bitmap, 657
touchgfx::PainterBW, 669	touchgfx::PainterARGB8888L8Bitmap, 660
touchgfx::PainterGRAY2, 675	touchgfx::PainterBGRA2222, 664
touchgfx::PainterGRAY2Bitmap, 678	touchgfx::PainterBGRA2222Bitmap, 667
touchgfx::PainterGRAY4, 681	touchgfx::PainterBWBitmap, 672
touchgfx::PainterGRAY4Bitmap, 684	touchgfx::PainterBW5ltmap, 672
•	_
touchgfx::PainterRGB565, 688	touchgfx::PainterGRAY2, 675
touchgfx::PainterRGB565Bitmap, 691	touchgfx::PainterGRAY2Bitmap, 678
touchgfx::PainterRGB565L8Bitmap, 694	touchgfx::PainterGRAY4, 681
touchgfx::PainterRGB888, 697	touchgfx::PainterGRAY4Bitmap, 685
touchgfx::PainterRGB888Bitmap, 702	touchgfx::PainterRGB565, 688
touchgfx::PainterRGB888L8Bitmap, 705	touchgfx::PainterRGB565Bitmap, 691
touchgfx::PainterRGBA2222, 708	touchgfx::PainterRGB565L8Bitmap, 695
touchgfx::PainterRGBA2222Bitmap, 711	touchgfx::PainterRGB888, 698
touchgfx::Rasterizer, 752	touchgfx::PainterRGB888Bitmap, 702
touchgfx::Renderer, 759	touchgfx::PainterRGB888L8Bitmap, 706
renderInit	touchgfx::PainterRGBA2222, 709
touchgfx::AbstractPainterABGR2222, 82	touchgfx::PainterRGBA2222Bitmap, 712

renderPixel	sampleKey
touchgfx::AbstractPainterABGR2222, 83	touchgfx::HALSDL2, 408
touchgfx::AbstractPainterARGB2222, 86	touchgfx::HAL, 396
touchgfx::AbstractPainterARGB8888, 88, 89	sampleTouch
touchgfx::AbstractPainterBGRA2222, 92	touchgfx::I2CTouchController, 414
touchgfx::AbstractPainterGRAY2, 96	touchgfx::NoTouchController, 632
touchgfx::AbstractPainterGRAY4, 98	touchgfx::SDL2TouchController, 828
touchgfx::AbstractPainterRGB565, 102	touchgfx::SDLTouchController, 829
touchgfx::AbstractPainterRGB888, 104	touchgfx::TouchController, 929
touchgfx::AbstractPainterRGBA2222, 107	saveNextScreenshots
Renderer, 758	touchgfx::HALSDL2, 408
touchgfx::Renderer, 759	saveScreenshot
RenderingAlgorithm	touchgfx::HALSDL2, 409
touchgfx::TextureMapper, 901	ScalableImage, 768
RenderingBuffer, 760	touchgfx::ScalableImage, 769
touchgfx::RenderingBuffer, 761	scaleTo24bpp
RenderingVariant	touchgfx::HALSDL2, 409
touchgfx, 54	ScalingAlgorithm
RepeatButton, 764	
touchgfx::RepeatButton, 764	touchgfx::ScalableImage, 769
RepeatButtonTrigger, 766	Scanline, 773
reportAsSolid	touchgfx::Scanline, 774
touchgfx::Box, 175	Scanline::iterator, 428
reset	Screen, 777
touchgfx::ButtonController, 188	touchgfx::Screen, 778
touchgfx::Outline, 638	ScrollBase, 794
touchgfx::Rasterizer, 752	touchgfx::ScrollBase, 797
touchgfx::Scanline, 777	ScrollList, 809
touchgfx::ScrollableContainer, 792	touchgfx::ScrollList, 810
resetDrawChainCache	ScrollWheel, 814
touchgfx::Drawable, 303	touchgfx::ScrollWheel, 815
resetExpandedStateTimer	ScrollWheelBase, 816
touchgfx::SlideMenu, 842	touchgfx::ScrollWheelBase, 817
resetSpans	ScrollWheelWithSelectionStyle, 820
touchgfx::Scanline, 777	touchgfx::ScrollWheelWithSelectionStyle, 821
resizeHeightToCurrentText	ScrollableContainer, 783
touchgfx::TextArea, 874	touchgfx::ScrollableContainer, 786
resizeToCurrentText	sdl_init
touchgfx::TextArea, 875	touchgfx::HALSDL2, 409
resizeToCurrentTextWithAlignment	seedExecution
touchgfx::TextArea, 875	touchgfx::DMA_Interface, 284
reverse	set
touchgfx::Vector, 957	touchgfx::Cell, 221
right	touchgfx::GPIO, 372
touchgfx::Rect, 758	setAction
rotate0	touchgfx::AbstractButton, 71
touchgfx, 56	touchgfx::AbstractButtonContainer, 72
rotate90	setAllowed
touchgfx, 56	touchgfx::DMA_Interface, 284
rotateRect	setAlpha
touchgfx::LCD, 456	touchgfx::AbstractButtonContainer, 72
row	touchgfx::Box, 174
touchgfx::RenderingBuffer, 763	touchgfx::BoxProgress, 177
	touchgfx::BoxWithBorder, 181
SDL2TouchController, 828	touchgfx::Button, 187
SDLTouchController, 829	touchgfx::CanvasWidget, 214
sample	touchgfx::CircleProgress, 239
touchgfx::ButtonController, 188	touchgfx::DigitalClock, 275
touchgfx::Buttons, 189	touchgfx::Image, 420

touchgfx::ImageProgress, 424	setBaselineY
touchgfx::LineProgress, 601	touchgfx::DigitalClock, 275
touchgfx::PainterABGR2222, 641	touchgfx::TextArea, 875
touchgfx::PainterABGR2222Bitmap, 645	setBitmap
touchgfx::PainterARGB2222, 648	touchgfx::AnimatedImage, 141
touchgfx::PainterARGB2222Bitmap, 651	touchgfx::Image, 420
touchgfx::PainterARGB8888, 654	touchgfx::ImageProgress, 425
touchgfx::PainterARGB8888Bitmap, 658	touchgfx::PainterABGR2222Bitmap, 645
touchgfx::PainterARGB8888L8Bitmap, 661	touchgfx::PainterARGB2222Bitmap, 651
touchgfx::PainterBGRA2222, 664	touchgfx::PainterARGB8888Bitmap, 658
touchgfx::PainterBGRA2222Bitmap, 667	touchgfx::PainterARGB8888L8Bitmap, 661
touchgfx::PainterGRAY2, 675	touchgfx::PainterBGRA2222Bitmap, 668
touchgfx::PainterGRAY2Bitmap, 679	touchgfx::PainterBWBitmap, 673
touchgfx::PainterGRAY4, 682	touchgfx::PainterGRAY2Bitmap, 679
touchgfx::PainterGRAY4Bitmap, 685	touchgfx::PainterGRAY4Bitmap, 685
touchgfx::PainterRGB565, 688	touchgfx::PainterRGB565Bitmap, 692
touchgfx::PainterRGB565Bitmap, 692	touchgfx::PainterRGB565L8Bitmap, 695
touchgfx::PainterRGB565L8Bitmap, 695	touchgfx::PainterRGB888Bitmap, 703
touchgfx::PainterRGB888, 698	touchgfx::PainterRGB888L8Bitmap, 706
touchgfx::PainterRGB888Bitmap, 703	touchgfx::PainterRGBA2222Bitmap, 713
touchgfx::PainterRGB888L8Bitmap, 706	touchgfx::ScalableImage, 772
touchgfx::PainterRGBA2222, 709	touchgfx::TextureMapper, 909
touchgfx::PainterRGBA2222Bitmap, 713	touchgfx::TiledImage, 916
touchgfx::PixelDataWidget, 719	setBitmapFormat
touchgfx::RadioButton, 742	touchgfx::PixeIDataWidget, 720
touchgfx::ScalableImage, 772	setBitmapPosition
touchgfx::SnapshotWidget, 863	touchgfx::TextureMapper, 909
touchgfx::TextArea, 875	setBitmapXY
touchgfx::TextProgress, 890	touchgfx::AnimatedImageButtonStyle, 144
touchgfx::TextureMapper, 909	touchgfx::ImageButtonStyle, 422
touchgfx::ZoomAnimationImage, 971	setBitmaps
setAnchorAtZero	touchgfx::AnimatedImage, 141
touchgfx::ImageProgress, 425	touchgfx::AnimatedImageButtonStyle, 144
setAngle	touchgfx::Button, 187
touchgfx::AbstractShape, 126	touchgfx::ButtonWithIcon, 191
setAnimateToCallback	touchgfx::ImageButtonStyle, 422
touchgfx::ScrollWheelBase, 819	touchgfx::RadioButton, 742
setAnimation	touchgfx::Slider, 851, 852
touchgfx::AnalogClock, 134	touchgfx::ToggleButton, 921
setAnimationDelay	touchgfx::ZoomAnimationImage, 971
touchgfx::ZoomAnimationImage, 971	setBorderColor
setAnimationDuration	touchgfx::BoxWithBorder, 181
touchgfx::SlideMenu, 843	setBorderSize
setAnimationEasingEquation	touchgfx::BoxWithBorder, 181
touchgfx::SlideMenu, 843	touchgfx::BoxWithBorderButtonStyle, 183
setAnimationEndedCallback	setBoxWithBorderColors
touchgfx::ScrollBase, 804	touchgfx::BoxWithBorderButtonStyle, 183
touchgfx::ZoomAnimationImage, 971	setBoxWithBorderHeight
setAnimationSteps	touchgfx::BoxWithBorderButtonStyle, 183
touchgfx::ScrollBase, 805	setBoxWithBorderPosition
setArc	touchgfx::BoxWithBorderButtonStyle, 184
touchgfx::Circle, 229	setBoxWithBorderWidth
setAuxiliaryLCD	touchgfx::BoxWithBorderButtonStyle, 184
touchgfx::HAL, 396	setBuffer
setBackground	touchgfx::Keyboard, 438
touchgfx::AbstractProgressIndicator, 118	setBufferPosition
touchgfx::AnalogClock, 135	touchgfx::Keyboard, 438
touchgfx::ModalWindow, 622	setButtonController

touchgfx::HAL, 396	setDebugPosition
setCCConsumed	touchgfx::DebugPrinter, 271
touchgfx::MCUInstrumentation, 618	setDebugPrinter
setCache	touchgfx::Application, 156
touchgfx::AbstractShape, 126	setDebugScale
touchgfx::Bitmap, 169	touchgfx::DebugPrinter, 272
touchgfx::Shape, 832	setDebugString
setCacheBitmap	touchgfx::Application, 156
touchgfx::CacheableContainer, 198	touchgfx::DebugPrinter, 272
setCalibrationMatrix	setDelay
touchgfx::TouchCalibration, 928	touchgfx::RepeatButton, 765
setCamera	touchgfx::RepeatButtonTrigger, 767
touchgfx::TextureMapper, 910	setDeselectedAction
setCameraDistance	touchgfx::RadioButton, 743
touchgfx::TextureMapper, 910	setDeselectionEnabled
setCapPrecision	touchgfx::RadioButton, 743
touchgfx::Circle, 230	touchgfx::RadioButtonGroup, 747
touchgfx::CircleProgress, 239	setDimension
touchgfx::Line, 591	touchgfx::ZoomAnimationImage, 972
setCenter	setDirection
touchgfx::Circle, 230, 231	touchgfx::AbstractDirectionProgress, 77
touchgfx::CircleProgress, 239	touchgfx::ListLayout, 606
setCircle	setDisplayMode
touchgfx::Circle, 231	touchgfx::DigitalClock, 276
setCircular	setDisplayOrientation
touchgfx::DrawableList, 312	touchgfx::HAL, 396
touchgfx::ScrollBase, 805	setDoneAction
touchgfx::ScrollWheelWithSelectionStyle, 823	touchgfx::AnimatedImage, 142
setClickAction	setDragAcceleration
touchgfx::ClickListener, 247	touchgfx::ScrollBase, 805
setColor	setDragAction
touchgfx::Box, 175	touchgfx::Snapper, 860
touchgfx::BoxProgress, 177	setDragThreshold
touchgfx::BoxWithBorder, 181	touchgfx::Gestures, 368
touchgfx::DigitalClock, 275	touchgfx::HAL, 397
touchgfx::PainterABGR2222, 642	setDrawableSize
touchgfx::PainterARGB2222, 648	touchgfx::DrawableList, 313
touchgfx::PainterARGB8888, 654	touchgfx::ScrollBase, 806
touchgfx::PainterBGRA2222, 664	touchgfx::ScrollWheelWithSelectionStyle, 824
touchgfx::PainterBW, 670	setDrawables
touchgfx::PainterGRAY2, 676	touchgfx::DrawableList, 313
touchgfx::PainterGRAY4, 682	touchgfx::ScrollList, 813
touchgfx::PainterRGB565, 689	touchgfx::ScrollWheel, 815
touchgfx::PainterRGB888, 700	touchgfx::ScrollWheelWithSelectionStyle, 824
touchgfx::PainterRGBA2222, 710	setEasingEquation
touchgfx::TextArea, 876	touchgfx::ScrollBase, 806
touchgfx::TextProgress, 890	setElement
setCoord	touchgfx::Matrix4x4, 615
touchgfx::Cell, 221	touchgfx::Quadruple, 737
setCorner	setEnd
touchgfx::AbstractShape, 127	touchgfx::Line, 591, 592
touchgfx::Shape, 833	touchgfx::LineProgress, 601
setCover	setEndSwipeElasticWidth
touchgfx::Cell, 222	touchgfx::SwipeContainer, 868
setCurrentState	setExpandDirection
touchgfx::ZoomAnimationImage, 972	touchgfx::SlideMenu, 843
setDebugColor	setExpandedStateTimeout
touchgfx::DebugPrinter, 271	touchgfx::SlideMenu, 843
todonginiDobagi filitor, 27 i	todongiaondolviona, o-to

setFadeAnimationDelay	setKeymappingList
touchgfx::FadeAnimator, 341	touchgfx::Keyboard, 439
setFadeAnimationEndedAction	setLabelColor
touchgfx::FadeAnimator, 342	touchgfx::ButtonWithLabel, 195
setFillingRule	setLabelColorPressed
touchgfx::Rasterizer, 752	touchgfx::ButtonWithLabel, 195
setFingerSize	setLabelRotation
touchgfx::HAL, 397	touchgfx::ButtonWithLabel, 196
setFontProvider	setLabelText
touchgfx::FontManager, 353	touchgfx::ButtonWithLabel, 196
setFrameBufferAllocator	setLanguage
touchgfx::HAL, 397	touchgfx::Texts, 896
setFrameBufferStartAddress	setLayout
touchgfx::HAL, 398	touchgfx::Keyboard, 439
setFrameBufferStartAddresses	setLine
touchgfx::HAL, 398	touchgfx::Line, 592
setFrameRateCompensation	setLineEndingStyle
touchgfx::HAL, 398	touchgfx::Line, 593
setFrameRefreshStrategy	touchgfx::LineProgress, 602
touchgfx::HAL, 399	setLineWidth
setHeight	touchgfx::Circle, 232
touchgfx::DigitalClock, 276	touchgfx::CircleProgress, 240
touchgfx::Drawable, 303	touchgfx::Line, 593, 594
touchgfx::DrawableList, 313	touchgfx::LineProgress, 602
touchgfx::ScrollBase, 806	setLinespacing
touchgfx::ScrollWheelWithSelectionStyle, 825	touchgfx::TextArea, 876
touchgfx::TiledImageButtonStyle, 918	setMCUActive
touchgfx::ZoomAnimationImage, 972	touchgfx::HAL, 399
setHiddenPixelsWhenExpanded	touchgfx::MCUInstrumentation, 618
touchgfx::SlideMenu, 843	setMCUInstrumentation
setHorizontal	touchgfx::HAL, 399
touchgfx::DrawableList, 314	setMaxRenderY
touchgfx::ScrollBase, 806	touchgfx::Outline, 638
touchgfx::ScrollWheelWithSelectionStyle, 825	touchgfx::Rasterizer, 753
setHourHandMinuteCorrection	setMaxSwipeItems
touchgfx::AnalogClock, 135	touchgfx::ScrollBase, 807
setIconBitmaps	setMaxVelocity
touchgfx::lconButtonStyle, 416	touchgfx::ScrollableContainer, 792
setIconXY	setMinuteHandSecondCorrection
touchgfx::ButtonWithIcon, 192	touchgfx::AnalogClock, 136
touchgfx::IconButtonStyle, 417	setMoveAnimationDelay
setIconX	touchgfx::MoveAnimator, 626
touchgfx::ButtonWithIcon, 192	setMoveAnimationEndedAction
touchgfx::lconButtonStyle, 417	touchgfx::MoveAnimator, 626
setIconY	setNewValueCallback
touchgfx::ButtonWithIcon, 192	touchgfx::Slider, 852
touchgfx::lconButtonStyle, 417	setNumberOfDecimals
setIndentation	touchgfx::TextProgress, 891
touchgfx::TextArea, 876	setNumberOfItems
setInterval	touchgfx::DrawableList, 314
touchgfx::RepeatButton, 766	touchgfx::ScrollBase, 808
touchgfx::RepeatButtonTrigger, 768	touchgfx::ScrollWheelWithSelectionStyle, 825
setItemPressedCallback	setOffset
touchgfx::ScrollBase, 807	touchgfx::AbstractPainter, 79
setItemSelectedCallback	touchgfx::DrawableList, 314
touchgfx::ScrollBase, 807	touchgfx::ScrollBase, 808
setKeyListener	touchgfx::ScrollWheelWithSelectionStyle, 826
touchgfx::Keyboard, 439	touchgfx::TiledImage, 916
todongiran oybodia, Too	todongian modifilago, oro

setOrigin	setScanlineWidth
touchgfx::AbstractShape, 127	touchgfx::CanvasWidgetRenderer, 219
setOrigo	setScreenContainer
touchgfx::TextureMapper, 910, 911	touchgfx::Transition, 931
setPadding	setScrollThreshold
touchgfx::ScrollList, 813	touchgfx::ScrollableContainer, 793
setPageIndicatorBitmaps	setScrollbarPadding
touchgfx::SwipeContainer, 868	touchgfx::ScrollableContainer, 792
setPageIndicatorXYWithCenteredX	setScrollbarWidth
touchgfx::SwipeContainer, 869	touchgfx::ScrollableContainer, 793
setPageIndicatorXY	setScrollbarsAlpha
touchgfx::SwipeContainer, 868	touchgfx::ScrollableContainer, 792
setPainter	setScrollbarsColor
touchgfx::CanvasWidget, 215	touchgfx::ScrollableContainer, 793
touchgfx::CircleProgress, 240	setScrollbarsPermanentlyVisible
touchgfx::LineProgress, 602	touchgfx::ScrollableContainer, 793
setPixelData	setScrollbarsVisible
touchgfx::PixelDataWidget, 720	touchgfx::ScrollableContainer, 793
setPosition	setSelected
touchgfx::Drawable, 303	touchgfx::RadioButton, 743
touchgfx::ZoomAnimationImage, 973	touchgfx::RadioButtonGroup, 749
setPrecision	setSelectedItemExtraSize
touchgfx::Circle, 232	touchgfx::ScrollWheelWithSelectionStyle, 826
setPressed	setSelectedItemMargin
touchgfx::AbstractButtonContainer, 73	touchgfx::ScrollWheelWithSelectionStyle, 826
setPressedAction	setSelectedItemOffset
touchgfx::TouchArea, 926	touchgfx::ScrollWheelBase, 819
setProgressIndicatorPosition	touchgfx::ScrollWheelWithSelectionStyle, 827
touchgfx::AbstractProgressIndicator, 118	setSelectedItemPosition
touchgfx::BoxProgress, 177	touchgfx::ScrollWheelWithSelectionStyle, 827
touchgfx::CircleProgress, 240	setSelectedPage
touchgfx::ImageProgress, 425	touchgfx::SwipeContainer, 869
touchgfx::LineProgress, 602	setShadeAlpha
touchgfx::TextProgress, 891	touchgfx::ModalWindow, 622
setRadioButtonDeselectedHandler	setShadeColor
touchgfx::RadioButtonGroup, 747	touchgfx::ModalWindow, 623
setRadioButtonSelectedHandler	setShape
touchgfx::RadioButtonGroup, 747	touchgfx::AbstractShape, 129
setRadius	setSnapPosition
touchgfx::Circle, 232	touchgfx::Snapper, 860
touchgfx::CircleProgress, 241	setSnappedAction
setRange	touchgfx::Snapper, 860
touchgfx::AbstractProgressIndicator, 119	setSnapping
setRenderingAlgorithm	touchgfx::ScrollList, 814
touchgfx::TextureMapper, 911	setStart
setRenderingBuffer	touchgfx::Line, 594, 595
touchgfx::Renderer, 759	touchgfx::LineProgress, 603
setRotation	setStartEndAngle
touchgfx::TextArea, 877	touchgfx::CircleProgress, 241
setRotationCenter	setStartValueCallback
touchgfx::AnalogClock, 136	touchgfx::Slider, 852
setScale	setState
touchgfx::AbstractShape, 128	touchgfx::SlideMenu, 844
touchgfx::TextureMapper, 911	set State Changed Animation Ended Callback
setScalingAlgorithm	touchgfx::SlideMenu, 844
touchgfx::ScalableImage, 772	setStateChangedCallback
setScalingMode	touchgfx::SlideMenu, 844
touchgfx::ZoomAnimationImage, 973	setStopValueCallback

touchgfx::Slider, 852	setTwoWildcardTextY
setSwipeAcceleration	touchgfx::TwoWildcardTextButtonStyle, 935
touchgfx::ScrollBase, 808	setType
setSwipeCutoff	touchgfx::ClickEvent, 245
touchgfx::SwipeContainer, 869	setTypedText
setTFTFrameBuffer	touchgfx::DigitalClock, 276
touchgfx::HALSDL2, 410	touchgfx::TextArea, 877
touchgfx::HAL, 400	touchgfx::TextProgress, 891
setText	setUpdateTicksInterval
touchgfx::TextButtonStyle, 886	touchgfx::AnimatedImage, 142
setTextColors	touchgfx::AnimatedImageButtonStyle, 144
touchgfx::TextButtonStyle, 886	setValue
setTextIndentation	touchgfx::AbstractProgressIndicator, 119
touchgfx::Keyboard, 439	touchgfx::BoxProgress, 178
setTextPosition	touchgfx::CircleProgress, 241
touchgfx::TextButtonStyle, 886	touchgfx::ImageProgress, 426
setTextRotation	touchgfx::LineProgress, 603
touchgfx::TextButtonStyle, 887	touchgfx::Slider, 854
setTextXY	touchgfx::TextProgress, 892
touchgfx::TextButtonStyle, 887	setValueRange
setTextureMapperAnimationEndedAction	touchgfx::Slider, 854, 855
touchgfx::AnimationTextureMapper, 148	setViewDistance
setTextureMapperAnimationStepAction	touchgfx::Matrix4x4, 616
touchgfx::AnimationTextureMapper, 148	setVisible
setTextX	touchgfx::Drawable, 304
touchgfx::TextButtonStyle, 887	setVisiblePixelsWhenCollapsed
setTextY	touchgfx::SlideMenu, 845
touchgfx::TextButtonStyle, 887	setVsyncInterval
setTileBitmaps	touchgfx::HALSDL2, 410
touchgfx::TiledImageButtonStyle, 918	setWideTextAction
setTileOffset	touchgfx::TextArea, 877
touchgfx::TiledImageButtonStyle, 919	setWidget
setTime12Hour	touchgfx::JSMOCHelper, 432
touchgfx::AbstractClock, 75	setWidgetAlpha
setTime24Hour	
touchgfx::AbstractClock, 75	touchgfx::AbstractPainter, 80 setWidth
-	
setToggleCanceled touchgfx::ToggleButtonTrigger, 924	touchgfx::DigitalClock, 276
	touchgfx::Drawable, 305 touchgfx::DrawableList, 315
setTop	touchgfx::ScrollBase, 809
touchgfx::GlyphNode, 370	-
setTouchSampleRate touchgfx::HAL, 400	touchgfx::ScrollWheelWithSelectionStyle, 827
•	touchgfx::TiledImageButtonStyle, 919
setTouchable	touchgfx::ZoomAnimationImage, 973
touchgfx::Drawable, 304	setWildcard
setTranslation	touchgfx::TextAreaWithOneWildcard, 880
touchgfx::Texts, 897	setWildcard1
setTwoWildcardText	touchgfx::TextAreaWithTwoWildcards, 883
touchgfx::TwoWildcardTextButtonStyle, 933	setWildcard2
setTwoWildcardTextColors	touchgfx::TextAreaWithTwoWildcards, 883
touchgfx::TwoWildcardTextButtonStyle, 933	setWildcardText
setTwoWildcardTextPosition	touchgfx::WildcardTextButtonStyle, 963
touchgfx::TwoWildcardTextButtonStyle, 934	setWildcardTextBuffer
setTwoWildcardTextRotation	touchgfx::WildcardTextButtonStyle, 963
touchgfx::TwoWildcardTextButtonStyle, 934	setWildcardTextBuffer1
setTwoWildcardTextXY	touchgfx::TwoWildcardTextButtonStyle, 935
touchgfx::TwoWildcardTextButtonStyle, 934	setWildcardTextBuffer2
setTwoWildcardTextX	touchgfx::TwoWildcardTextButtonStyle, 935
touchgfx::TwoWildcardTextButtonStyle, 934	setWildcardTextColors

touchgfx::WildcardTextButtonStyle, 964	setW
setWildcardTextPosition	touchgfx::Quadruple, 737
touchgfx::WildcardTextButtonStyle, 964	setX
setWildcardTextRotation	touchgfx::ClickEvent, 245
touchgfx::WildcardTextButtonStyle, 964	touchgfx::Drawable, 305
setWildcardTextXY	touchgfx::Quadruple, 738
touchgfx::WildcardTextButtonStyle, 965	setY
setWildcardTextX	touchgfx::ClickEvent, 246
touchgfx::WildcardTextButtonStyle, 964	touchgfx::Drawable, 306
setWildcardTextY	touchgfx::Quadruple, 738
touchgfx::WildcardTextButtonStyle, 965	setZ
setWindowSize	
	touchgfx::Quadruple, 738
touchgfx::ScrollList, 814	Shape < POINTS >, 830
setWindowTitle	show
touchgfx::HALSDL2, 410	touchgfx::ModalWindow, 623
setWriteMemoryUsageReport	signalDMAInterrupt
touchgfx::CanvasWidgetRenderer, 219	touchgfx::DMA_Interface, 285
setXBaselineY	touchgfx::HAL, 400
touchgfx::TextArea, 877	touchgfx::NoDMA, 631
setXOffset	signalVSync
touchgfx::TiledImage, 916	touchgfx::OSWrappers, 634
setXY	sine
touchgfx::Drawable, 305	touchgfx::CWRUtil, 268
setYOffset	sineEaseIn
touchgfx::TiledImage, 917	touchgfx::EasingEquations, 334
setup	sineEaseInOut
touchgfx::SlideMenu, 844, 845	touchgfx::EasingEquations, 334
setupAnimation	sineEaseOut
touchgfx::AnimationTextureMapper, 148	touchgfx::EasingEquations, 335
setupBuffer	SingleBlockAllocator< block_size, bytes_pr_pixel >,
touchgfx::CanvasWidgetRenderer, 219	834
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy	834 size
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284	834 size touchgfx::Vector, 957
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631	834 size touchgfx::Vector, 957 SlideMenu, 836
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill	834 size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631	834 size touchgfx::Vector, 957 SlideMenu, 836
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill	834 size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285	834 size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631	834 size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain	834 size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition SlideTransition SlideTransition SlideTransition
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$856 SlideTransition \$856 SlideTransition \$856 SlideRough
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper touchgfx::Snapper, 859 Snapper< T >, 858
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Keyboard, 440	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper touchgfx::Snapper, 859 Snapper< T >, 858 SnapshotWidget, 861
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Keyboard, 440 touchgfx::PreRenderable, 724	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 SlideTransition SlideTransition \$856 Snapper touchgfx::Snapper, 859 SnapshotWidget, 861 touchgfx::SnapshotWidget, 862
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Keyboard, 440 touchgfx::PreRenderable, 724 setupHand	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper touchgfx::Snapper, 859 Snapper< T >, 858 SnapshotWidget, 861 touchgfx::SnapshotWidget, 862 snprintf
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Reyboard, 440 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper touchgfx::Snapper, 859 Snapper< T >, 858 SnapshotWidget, 861 touchgfx::SnapshotWidget, 862 snprintf touchgfx::Unicode, 944
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Keyboard, 440 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$50 Touchgfx::SlideTransition \$57 SlideTra
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::Reyboard, 440 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition< templateDirection >, 856 SlideTransition SlideTransition \$856 Snapper touchgfx::Snapper, 859 Snapper \$7 touchgfx::SnapshotWidget, 862 snprintff touchgfx::Unicode, 944 snprintfFloat touchgfx::Unicode, 945, 946
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$56 Snapper \$56 Snapper \$56
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableCo	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition 856 SlideTransitio
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::Heyboard, 440 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$57 SlideTransition \$58 SlideTransition \$56 Snapper \$59 Snapper \$59 Snapper \$59 Snapper \$59 <t< td=""></t<>
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$59 SnapshotWidget, 861 \$62 touchgfx::Unicode, 944 \$62 snprintfFloat \$50 touchgfx::Unicode, 945, 946 snprintfFloats \$50 touchgfx::Unicode, 947, 948 sqrtQ10 \$50 touchgfx::CWRUtil, 268
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137 setupScreen	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$59 Snapper \$59 SnapphotWidget, 861 \$62 snprintf touchgfx::SnapshotWidget, 862 snprintfFloat touchgfx::Unicode, 944 snprintfFloats touchgfx::Unicode, 945, 946 snprintfFloats touchgfx::Unicode, 947, 948 sqrtQ10 touchgfx::CWRUtil, 268 srcFormat
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$59 SnapshotWidget, 861 \$62 touchgfx::Unicode, 944 \$62 snprintfFloat \$50 touchgfx::Unicode, 945, 946 snprintfFloats \$50 touchgfx::Unicode, 947, 948 sqrtQ10 \$50 touchgfx::CWRUtil, 268
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Container, 259 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137 setupScreen	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$59 Snapper \$59 SnapphotWidget, 861 \$62 snprintf touchgfx::SnapshotWidget, 862 snprintfFloat touchgfx::Unicode, 944 snprintfFloats touchgfx::Unicode, 945, 946 snprintfFloats touchgfx::Unicode, 947, 948 sqrtQ10 touchgfx::CWRUtil, 268 srcFormat
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137 setupScreen touchgfx::Screen, 782	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition touchgfx::SlideTransition, 857 SlideTransition templateDirection >, 856 SlideTransition templateDirection >, 856 Slider, 846 touchgfx::Slider, 848 Snapper touchgfx::Snapper, 859 Snapper touchgfx::Snapper, 859 SnapshotWidget, 861 touchgfx::SnapshotWidget, 862 snprintf touchgfx::Unicode, 944 snprintfFloat touchgfx::Unicode, 944 snprintfFloats touchgfx::Unicode, 945, 946 snprintfFloats touchgfx::Unicode, 947, 948 sqrtQ10 touchgfx::CWRUtil, 268 srcFormat touchgfx::BlitOp, 171
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137 setupScreen touchgfx::Screen, 782 setupSecondHand	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition 856 Snapper 859 Snapper 859 Snapper 859 Snapper 862 SnapshotWidget, 861 touchgfx::SnapshotWidget, 862 Snprintf touchgfx::Unicode, 944 SnprintfFloat touchgfx::Unicode, 945, 946 SnprintfFloats touchgfx::Unicode, 945, 946 SnprintfFloats touchgfx::Unicode, 947, 948 SqrtQ10 touchgfx::CWRUtil, 268 SrcFormat touchgfx::BlitOp, 171 Start
touchgfx::CanvasWidgetRenderer, 219 setupDataCopy touchgfx::DMA_Interface, 284 touchgfx::NoDMA, 631 setupDataFill touchgfx::DMA_Interface, 285 touchgfx::NoDMA, 631 setupDrawChain touchgfx::CacheableContainer, 199 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::CacheableContainer::CachedImage, 200 touchgfx::Drawable, 304 touchgfx::Drawable, 304 touchgfx::PreRenderable, 724 setupHand touchgfx::AnalogClock, 136 setupHorizontalSlider touchgfx::Slider, 853 setupHourHand touchgfx::AnalogClock, 137 setupMinuteHand touchgfx::AnalogClock, 137 setupScreen touchgfx::Screen, 782 setupSecondHand touchgfx::AnalogClock, 137	size touchgfx::Vector, 957 SlideMenu, 836 SlideTransition touchgfx::SlideTransition, 857 SlideTransition SlideTransition \$56 Snapper \$59 Snapper \$70 touchgfx::Snapper, 859 \$62 Snapper \$70 touchgfx::Snapper, 859 \$62 Snapper \$70 touchgfx::Unicode, 944 \$62 snprintfFloat \$70 touchgfx::Unicode, 945, 946 snprintfFloats \$70 touchgfx::Unicode, 947, 948 sqrtQ10 \$70 touchgfx::CWRUtil, 268 srcFormat \$70 touchgfx::BlitOp, 171 start \$70 touchgfx::DMA_Interface, 285

touchgfx::AnimationTextureMapper, 149	touchgfx::Screen, 782
startFadeAnimation	TextArea, 870
touchgfx::FadeAnimator, 342	touchgfx::TextArea, 871
startFlashLineRead	TextAreaWithOneWildcard, 878
touchgfx::FlashDataReader, 343	touchgfx::TextAreaWithOneWildcard, 879
startMoveAnimation	TextAreaWithTwoWildcards, 880
touchgfx::MoveAnimator, 626	touchgfx::TextAreaWithTwoWildcards, 881
startSMOC	TextAreaWithWildcardBase, 883
touchgfx::Screen, 782	touchgfx::TextAreaWithWildcardBase, 884
startTimerAndSetParameters	TextButtonStyle < T >, 885
touchgfx::ZoomAnimationImage, 973	TextDirection
startZoomAndMoveAnimation	touchgfx, 55
touchgfx::ZoomAnimationImage, 974	TextProgress, 888
startZoomAnimation	touchgfx::TextProgress, 889
touchgfx::ZoomAnimationImage, 975	TextProvider, 892
stateChangeButtonClickedHandler	touchgfx::TextProvider, 893
touchgfx::SlideMenu, 845	TextRotation
States	touchgfx, 57
touchgfx::ZoomAnimationImage, 968	Texts, 895
step	
touchgfx::Edge, 337	TextureSurface, 012
stopAnimation	TextureSurface, 912
touchgfx::AnimatedImage, 142	tick
touchgfx::ScrollBase, 809	touchgfx::Gestures, 368
StringVisuals	touchgfx::HAL, 401
touchgfx::LCD::StringVisuals, 864, 865	tickMoveDrawable
stringWidth	touchgfx::CoverTransition, 262
touchgfx::LCD, 457	touchgfx::SlideTransition, 858
	TiledImage, 913
strlen	touchgfx::TiledImage, 914
touchgfx::Unicode, 949, 950	TiledImageButtonStyle< T >, 917
strncmp	to
touchgfx::Unicode, 950	touchgfx::CWRUtil::Q10, 728
strncmp_ignore_white_spaces	touchgfx::CWRUtil::Q5, 734
touchgfx::Unicode, 951	toQ10
strncpy	touchgfx::CWRUtil, 269
touchgfx::Unicode, 951	toQ5
SupportedTypesList	touchgfx::CWRUtil, 269
touchgfx::Partition, 716	toUTF8
swapFrameBuffers	touchgfx::Unicode, 952
touchgfx::HAL, 400	toggle
SwipeContainer, 865	touchgfx::GPIO, 372
switchScreen	touchgfx::LED, 586
touchgfx::Application, 156	ToggleButton, 919
TYPED_TEXT_INVALID	touchgfx::ToggleButton, 920
touchgfx, 68	ToggleButtonTrigger, 922
takeFrameBufferSemaphore	top
·	touchgfx::GlyphNode, 370
touchgfx::OSWrappers, 635	touch
taskDelay	touchgfx::HAL, 401
touchgfx::HAL, 400	_
touchgfx::OSWrappers, 635	TouchArea, 924
taskEntry	touchgfx::TouchArea, 925
touchgfx::HALSDL2, 411	TouchButtonTrigger, 927
touchgfx::HAL, 401	TouchCalibration, 928
tearDown	TouchController, 928
touchgfx::CoverTransition, 262	touchgfx, 41
touchgfx::SlideTransition, 858	abs, 58
touchgfx::Transition, 931	Alignment, 54
tearDownScreen	BlitOperations, 55

ceil28_4, 58	getCurrentSecond, 74
clz, 59	setTime12Hour, 75
DMAType, 56	setTime24Hour, 75
Direction, 55	updateClock, 75
DisplayOrientation, 56	touchgfx::AbstractDirectionProgress
DisplayRotation, 56	\sim AbstractDirectionProgress, 77
EasingEquation, 54	AbstractDirectionProgress, 76
finalizeTransition, 59	DirectionType, 76
fixed28_4Mul, 59	getDirection, 77
fixed28_4ToFloat, 60	setDirection, 77
floatToFixed16_16, 60	touchgfx::AbstractPainter
floatToFixed28_4, 60	~AbstractPainter, 78
floorDivMod, 60	AbstractPainter, 78
FrameBuffer, 56	compatibleFramebuffer, 78
FrameBufferAllocatorSignalBlockDrawn, 61	render, 79
FrameBufferAllocatorWaitOnTransfer, 61	setOffset, 79
gcd, 61	setWidgetAlpha, 80
GlyphFlags, 57	touchgfx::AbstractPainterABGR2222
Gradient, 57	mixColors, 81
hw_init, 61	render, 82
LCD2getPixel, 61, 62	
LCD2setPixel, 62	renderlnit, 82
•	renderNext, 82
LCD2shiftVal, 63	renderPixel, 83
LCD4getPixel, 63	touchgfx::AbstractPainterARGB2222
LCD4setPixel, 64	mixColors, 84
lookupBilinearRenderVariant, 64	render, 85
lookupNearestNeighborRenderVariant, 64	renderInit, 85
makeTransition, 65	renderNext, 86
memset, 65	renderPixel, 86
muldiv, 66	touchgfx::AbstractPainterARGB8888
operator*, 66	render, 87
prepareTransition, 67	renderInit, 88
RenderingVariant, 54	renderNext, 88
rotate0, 56	renderPixel, 88, 89
rotate90, 56	touchgfx::AbstractPainterBGRA2222
TYPED_TEXT_INVALID, 68	mixColors, 90
TextDirection, 55	render, 91
TextRotation, 57	renderInit, 91
touchgfx_generic_init, 67	renderNext, 92
touchgfx_init, 68	renderPixel, 92
WideTextAction, 57	touchgfx::AbstractPainterBW
touchgfx::AbstractButton	render, 93
AbstractButton, 70	renderInit, 94
getPressedState, 70	renderNext, 94
getType, 70	touchgfx::AbstractPainterGRAY2
handleClickEvent, 70	render, 95
setAction, 71	renderInit, 95
touchgfx::AbstractButtonContainer	renderNext, 96
getAlpha, 72	renderPixel, 96
getPressed, 72	touchgfx::AbstractPainterGRAY4
setAction, 72	render, 97
setAlpha, 72	renderInit, 98
setPressed, 73	renderNext, 98
touchgfx::AbstractClock	renderPixel, 98
~AbstractClock, 74	touchgfx::AbstractPainterRGB565
AbstractClock, 74	mixColors, 100
getCurrentHour, 74	render, 101
getCurrentMinute, 74	renderly 101
gotouriontivinialo, / T	rondomin, ron

renderNext, 101	setOrigin, 127
renderPixel, 102	setScale, 128
touchgfx::AbstractPainterRGB888	setShape, 129
render, 103	updateAbstractShapeCache, 129
renderInit, 103	updateAngle, 129
renderNext, 104	updateScale, 130
renderPixel, 104	touchgfx::AnalogClock
touchgfx::AbstractPainterRGBA2222	animationEnabled, 132
mixColors, 105, 106	convertHandValueToAngle, 132
render, 106	getAnimationDuration, 133
renderInit, 107	getHourHandMinuteCorrection, 133
renderNext, 107	getMinuteHandSecondCorrection, 133
renderPixel, 107	initializeTime12Hour, 134
touchgfx::AbstractPartition	initializeTime24Hour, 134
~AbstractPartition, 109	setAnimation, 134
AbstractPartition, 109	setBackground, 135
allocate, 109, 110	setHourHandMinuteCorrection, 135
allocateAt, 110	setMinuteHandSecondCorrection, 136
at, 111	setRotationCenter, 136
capacity, 112	setupHand, 136
clear, 112	setupHourHand, 137
dec, 112	setupMinuteHand, 137
element, 112	setupSecondHand, 137
element_size, 113	updateClock, 138
find, 113	touchgfx::AnimatedImage
getAllocationCount, 113	AnimatedImage, 139, 140
indexOf, 114	getType, 140
touchgfx::AbstractProgressIndicator	handleTickEvent, 140
~AbstractProgressIndicator, 116	isAnimatedImageRunning, 140
AbstractProgressIndicator, 115	isReverse, 141
getProgress, 116	isRunning, 141
getProgressIndicatorHeight, 116	pauseAnimation, 141
getProgressIndicatorWidth, 116	setBitmap, 141
getProgressIndicatorX, 117	setBitmaps, 141
getProgressIndicatorY, 117	setDoneAction, 142
getRange, 117, 118	setUpdateTicksInterval, 142
getValue, 118	startAnimation, 142
setBackground, 118	stopAnimation, 142
setProgressIndicatorPosition, 118	touchgfx::AnimatedImageButtonStyle
setRange, 119	setBitmapXY, 144
setValue, 119	setBitmaps, 144
touchgfx::AbstractShape	setUpdateTicksInterval, 144
~AbstractShape, 122	touchgfx::AnimationTextureMapper
AbstractShape, 121	~AnimationTextureMapper, 147
drawCanvasWidget, 122	AnimationParameter, 146
getAngle, 122	AnimationState, 147
getCacheX, 123	AnimationTextureMapper, 147
getCacheY, 123	cancelAnimationTextureMapperAnimation, 147
getCornerX, 123	getAnimationStep, 147
getCornerY, 124	handleTickEvent, 147
getMinimalRect, 124	isTextureMapperAnimationRunning, 148
getNumPoints, 124	setTextureMapperAnimationEndedAction, 148
getOrigin, 124	setTextureMapperAnimationStepAction, 148
getScale, 125	setupAnimation, 148
moveOrigin, 125	startAnimation, 149
setAngle, 126	touchgfx::Application
setCache, 126	appSwitchScreen, 151
setCorner, 127	Application, 151
	rr /

and a Dynam On a vertice at 150	to colo of colDitOn
cacheDrawOperations, 152	touchgfx::BlitOp
clearAllTimerWidgets, 152	dstFormat, 171
draw, 152	operation, 171
getCurrentScreen, 152	srcFormat, 171
getDebugPrinter, 153	touchgfx::Box
getInstance, 153	\sim Box, 173
getNumberOfRegisteredTimerWidgets, 153	Box, 172
getTimerWidgetCountForDrawable, 153	draw, 173
handleClickEvent, 154	forceReportAsSolid, 173
handleDragEvent, 154	getAlpha, 174
handleGestureEvent, 154	getColor, 174
handleKeyEvent, 154	getSolidRect, 174
handlePendingScreenTransition, 155	getType, 174
handleTickEvent, 155	reportAsSolid, 175
instance, 157	setAlpha, 174
invalidateArea, 155	setColor, 175
MAX_TIMER_WIDGETS, 157	touchgfx::BoxProgress
registerTimerWidget, 155	~BoxProgress, 176
setDebugPrinter, 156	BoxProgress, 176
setDebugString, 156	getAlpha, 176
switchScreen, 156	getColor, 177
unregisterTimerWidget, 156	setAlpha, 177
touchgfx::Bitmap	setColor, 177
Bitmap, 160	setProgressIndicatorPosition, 177
BitmapFormat, 159	setValue, 178
•	
cache, 160	touchgfx::BoxWithBorder
cacheAll, 161	BoxWithBorder, 179
cacheGetAddress, 161	draw, 180
cachelsCached, 161	getAlpha, 180
cacheRemoveBitmap, 162	getBorderColor, 180
cacheReplaceBitmap, 162	getBorderSize, 180
clearCache, 162	getColor, 180
ClutFormat, 159	getSolidRect, 180
compactCache, 162	getType, 181
dynamicBitmapAddSolidRect, 163	setAlpha, 181
dynamicBitmapCreate, 163	setBorderColor, 181
dynamicBitmapDelete, 164	setBorderSize, 181
dynamicBitmapGetAddress, 164	setColor, 181
dynamicBitmapSetSolidRect, 164	touchgfx::BoxWithBorderButtonStyle
getAlphaData, 165	setBorderSize, 183
getCacheTopAddress, 165	setBoxWithBorderColors, 183
getData, 165	setBoxWithBorderHeight, 183
getExtraData, 166	setBoxWithBorderPosition, 184
getFormat, 166	setBoxWithBorderWidth, 184
getHeight, 166	touchgfx::Button
getld, 166	\sim Button, 185
getRect, 166	Button, 185
getSolidRect, 167	draw, 186
getWidth, 167	getAlpha, 186
hasTransparentPixels, 167	getCurrentlyDisplayedBitmap, 186
isAlphaPerPixel, 167	getSolidRect, 186
operator!=, 167	getType, 186
operator==, 168	setAlpha, 187
registerBitmapDatabase, 168	setBitmaps, 187
removeCache, 168	touchgfx::ButtonController
setCache, 169	~ButtonController, 188
touchgfx::Bitmap::BitmapData	init, 188
getFormat, 170	reset, 188
gon orman, 170	10001, 100

sample, 188	to, 734
touchgfx::ButtonWithIcon	touchgfx::CacheableContainer
draw, 190	\sim CacheableContainer, 197
getCurrentlyDisplayedIcon, 190	CacheableContainer, 197
getlconX, 191	enableCachedMode, 198
getlconY, 191	invalidateRect, 198
getType, 191	isChildInvalidated, 198
setBitmaps, 191	setCacheBitmap, 198
setIconXY, 192	setupDrawChain, 199
setIconX, 192	updateCache, 199
setIconY, 192	touchgfx::CacheableContainer::CachedImage
touchgfx::ButtonWithLabel	~CachedImage, 200
ButtonWithLabel, 194	CachedImage, 200
draw, 194	setupDrawChain, 200
getLabelRotation, 194	touchgfx::Callback
getLabelText, 194	Callback, 202
-	
getSolidRect, 195	execute, 202
getType, 195	isValid, 202
setLabelColor, 195	touchgfx::Callback< dest_type, T1, T2, void >
setLabelColorPressed, 195	Callback, 203, 204
setLabelRotation, 196	execute, 204
setLabelText, 196	isValid, 204
updateTextPosition, 196	touchgfx::Callback< dest_type, T1, void, void >
touchgfx::Buttons	Callback, 205
init, 189	execute, 206
sample, 189	isValid, 206
touchgfx::CWRUtil	touchgfx::Callback< dest_type, void, void, void >
angle, 264, 265	Callback, 207
arcsine, 265	execute, 208
cosine, 265, 266	isValid, 208
mulQ5, 267	touchgfx::Canvas
muldivQ10, 266	\sim Canvas, 209
muldivQ5, 266	Canvas, 209
sine, 268	lineTo, 209, 210
sqrtQ10, 268	moveTo, 210, 211
toQ10, 269	render, 211
toQ5, 269	touchgfx::CanvasWidget
touchgfx::CWRUtil::Q10	~CanvasWidget, 212
operator int, 726	CanvasWidget, 212
operator*, 726	draw, 212
operator+, 727	drawCanvasWidget, 213
operator-, 727	getAlpha, 213
operator/, 727	getMinimalRect, 213
Q10, 726	getPainter, 214
to, 728	getSolidRect, 214
touchgfx::CWRUtil::Q15	invalidate, 214
operator int, 729	setAlpha, 214
operator+, 729	setPainter, 215
operator, 729	touchgfx::CanvasWidgetRenderer
operator/, 730	getMissingBufferSize, 216
Q15, 729	getOutlineBuffer, 216
touchgfx::CWRUtil::Q5	getOutlineBufferSize, 217
operator int, 731	getScanlineCounts, 217
operator*, 732	getScanlineCovers, 217
operator+, 733	getScanlineStartIndices, 217
operator-, 733	getScanlineWidth, 217
operator/, 734	getUsedBufferSize, 218
Q5, 731	getWriteMemoryUsageReport, 218

hasBuffer, 218	ClickEventType, 243
numCellsMissing, 218	getEventType, 244
numCellsUsed, 218	getForce, 244
setScanlineWidth, 219	getType, 244
setWriteMemoryUsageReport, 219	getX, 245
setupBuffer, 219	getY, 245
touchgfx::Cell	setType, 245
addCover, 221	setX, 245
packedCoord, 221	setY, 246
set, 221	touchgfx::ClickListener
setCoord, 221	ClickListener, 246
setCover, 222	handleClickEvent, 247
touchgfx::Circle	setClickAction, 247
Circle, 224	touchgfx::Color
drawCanvasWidget, 224	getBlueColor, 248
getArc, 224	getColorFrom24BitRGB, 248
getArcEnd, 225	getGreenColor, 248
getArcStart, 225, 226	getRedColor, 249
getCapPrecision, 226	touchgfx::ConstFont
getCenter, 226	ConstFont, 252
getLineWidth, 227	find, 252
getMinimalRect, 227, 228	getGlyph, 252
getPrecision, 228	getKerning, 253
getRadius, 228	getPixelData, 253
setArc, 229	touchgfx::Container
setCapPrecision, 230	~Container, 255
setCenter, 230, 231	add, 255
setCircle, 231	Container, 255
setLineWidth, 232	contains, 256
setPrecision, 232	draw, 256
setRadius, 232	forEachChild, 256
updateArc, 233	getContainedArea, 256
updateArcEnd, 233	getFirstChild, 257
updateArcStart, 235	getLastChild, 257
touchgfx::CircleProgress	getSolidRect, 257
~CircleProgress, 237	getType, 258
CircleProgress, 237	insert, 258
getAlpha, 237	moveChildrenRelative, 258
getCapPrecision, 237	remove, 259
getCenter, 237	removeAll, 259
getEndAngle, 238	setupDrawChain, 259
getLineWidth, 238	unlink, 259
getRadius, 238	touchgfx::CoverTransition
getStartAngle, 238	~CoverTransition, 261
-	
setAlpha, 239 setCapPrecision, 239	CoverTransition, 261
•	handleTickEvent, 261
setCenter, 239	init, 261
setLineWidth, 240	initMoveDrawable, 261
setPainter, 240	tearDown, 262
setProgressIndicatorPosition, 240	tickMoveDrawable, 262
setRadius, 241	touchgfx::CoverTransition::FullSolidRect
setStartEndAngle, 241	draw, 357
setValue, 241	getSolidRect, 357
touchgfx::ClickButtonTrigger	touchgfx::DMA_Interface
handleClickEvent, 242	~DMA_Interface, 281
touchgfx::ClickEvent	addToQueue, 281
~ClickEvent, 244	DMA_Interface, 281
ClickEvent, 243	disableAlpha, 281

enableAlpha, 282	getEventType, 289
enableCopyWithTransparentPixels, 282	getNewX, 289
execute, 282	getNewY, 290
executeCompleted, 282	getOldX, 290
flush, 282	getOldY, 290
getAllowed, 282	getType, 290
getBlitCaps, 283	touchgfx::Draggable
getDMAType, 283	\sim Draggable, 291
initialize, 283	Draggable, 291
isDMARunning, 284	handleDragEvent, 291
isDmaQueueEmpty, 283	touchgfx::Drawable
isDmaQueueFull, 283	~Drawable, 295
seedExecution, 284	childGeometryChanged, 295
setAllowed, 284	draw, 295
setupDataCopy, 284	drawToDynamicBitmap, 296
setupDataFill, 285	Drawable, 295
signalDMAInterrupt, 285	DrawableType, 295
start, 285	getAbsoluteRect, 296
waitForFrameBufferSemaphore, 285	getCachedAbsX, 296
touchgfx::DMA Queue	_
_	getCachedAbsY, 296
~DMA_Queue, 286	getCachedVisibleRect, 296
DMA_Queue, 286	getHeight, 297
first, 286	getLastChild, 297
isEmpty, 286	getNextSibling, 297
isFull, 287	getParent, 298
pop, 287	getRect, 298
pushCopyOf, 287	getSolidRect, 298
touchgfx::DebugPrinter	getSolidRectAbsolute, 298
draw, 270	getType, 299
region, 271	getVisibleRect, 299
setDebugColor, 271	getWidth, 299
setDebugPosition, 271	getX, 299
setDebugScale, 272	getY, 300
setDebugString, 272	handleClickEvent, 300
touchgfx::DigitalClock	handleDragEvent, 300
\sim DigitalClock, 274	handleGestureEvent, 300
DigitalClock, 274	handleTickEvent, 301
displayLeadingZeroForHourIndicator, 274	invalidate, 301
DisplayMode, 273	invalidateRect, 301
getAlpha, 274	isTouchable, 302
getDisplayMode, 274	isVisible, 302
getTextWidth, 275	moveRelative, 302
setAlpha, 275	moveTo, 302
setBaselineY, 275	resetDrawChainCache, 303
setColor, 275	setHeight, 303
setDisplayMode, 276	setPosition, 303
setHeight, 276	setTouchable, 304
setTypedText, 276	setVisible, 304
setWidth, 276	setWidth, 305
updateClock, 277	setXY, 305
touchgfx::DisplayTransformation	setupDrawChain, 304
transformDisplayToFrameBuffer, 277–279	setX, 305
transformFrameBufferToDisplay, 279	setY, 306
touchgfx::DragEvent	translateRectToAbsolute, 306
\sim DragEvent, 289	touchgfx::DrawableList
DragEvent, 288	∼DrawableList, 308
getDeltaX, 289	DrawableList, 308
getDeltaY, 289	getCircular, 308
3 ,	3,

getDrawableIndex, 308	quartEaseInOut, 332
getDrawableIndices, 309	quartEaseOut, 332
getDrawableMargin, 309	quintEaseIn, 333
getDrawableSize, 309	quintEaseInOut, 333
getHorizontal, 310	quintEaseOut, 334
getItemIndex, 310	sineEaseIn, 334
getItemSize, 310	sineEaseInOut, 334
getNumberOfDrawables, 311	sineEaseOut, 335
getNumberOfItems, 311	touchgfx::Edge
getOffset, 311	Edge, 336
getRequiredNumberOfDrawables, 311	step, 337
itemChanged, 312	touchgfx::Event
refreshDrawables, 312	\sim Event, 338
setCircular, 312	EventType, 338
setDrawableSize, 313	getEventType, 338
setDrawables, 313	touchgfx::FadeAnimator
setHeight, 313	\sim FadeAnimator, 340
setHorizontal, 314	clearFadeAnimationEndedAction, 340
setNumberOfItems, 314	FadeAnimator, 340
setOffset, 314	getFadeAnimationDelay, 340
setWidth, 315	handleTickEvent, 341
touchgfx::DrawableListItems	isFadeAnimationRunning, 341
~DrawableListItems, 316	isRunning, 341
DrawableListItems, 316	nextFadeAnimationStep, 341
getDrawable, 316	setFadeAnimationDelay, 341
getNumberOfDrawables, 317	setFadeAnimationEndedAction, 342
operator[], 317	startFadeAnimation, 342
touchgfx::DrawableListItemsInterface	touchgfx::FlashDataReader
~DrawableListItemsInterface, 318	addresslsAddressable, 343
getDrawable, 318	copyData, 343
getNumberOfDrawables, 318	startFlashLineRead, 343
touchgfx::EasingEquations	waitFlashReadComplete, 344
backEaseIn, 321	touchgfx::Font
backEaseInOut, 321	Font, 346
backEaseOut, 322	getBitsPerPixel, 346
bounceEaseIn, 322	getCharWidth, 347
bounceEaseInOut, 323	getDataFormatA4, 347
bounceEaseOut, 323	getEllipsisChar, 347
circEaseIn, 323	getFallbackChar, 347
circEaseInOut, 324	getFontHeight, 347
circEaseOut, 324	getGSUBTable, 349
cubicEaseIn, 325	getGlyph, 348
cubicEaseInOut, 325	getKerning, 349
cubicEaseOut, 326	getMaxPixelsLeft, 349
elasticEaseIn, 326	getMaxPixelsRight, 349
elasticEaseInOut, 326	getMaxTextHeight, 350
elasticEaseOut, 327	getMinimumTextHeight, 350
expoEaseIn, 327	getNumberOfLines, 350
expoEaseInOut, 328	getSpacingAbove, 351
expoEaseOut, 328	getStringWidth, 351
linearEaseIn, 329	getStringWidthLTR, 352
linearEaseInOut, 329	getStringWidthRTL, 352
linearEaseNone, 329	touchgfx::FontManager
linearEaseOut, 330	getFont, 353
quadEaseIn, 330	setFontProvider, 353
quadEaseInOut, 331	touchgfx::FontProvider
quadEaseOut, 331	~FontProvider, 354
quartEaseIn, 331	getFont, 354
4-art-400mi, 001	900 000

touchgfx::FrameBufferAllocator	copyScreenshotToClipboard, 404
allocateBlock, 355	disableInterrupts, 405
freeBlockAfterTransfer, 356	doRotate, 405
getBlockForTransfer, 356	doSampleTouch, 405
hasBlockReadyForTransfer, 356	enableInterrupts, 405
markBlockReadyForTransfer, 356	enableLCDControllerInterrupt, 406
touchgfx::GPIO	flushFrameBuffer, 406
clear, 371	getArgv, 406
GPIO_ID, 371	getTFTFrameBuffer, 407
get, 372	getWindowTitle, 407
init, 372	HALSDL2, 403
set, 372	loadSkin, 407
toggle, 372	performDisplayOrientationChange, 408
touchgfx::GenericCallback	renderLCD_FrameBufferToMemory, 408
~GenericCallback, 358	sampleKey, 408
execute, 359	saveNextScreenshots, 408
isValid, 359	saveScreenshot, 409
touchgfx::GenericCallback< T1, T2, void >	scaleTo24bpp, 409
~GenericCallback, 360	sdl init, 409
execute, 360	setTFTFrameBuffer, 410
isValid, 361	setVsyncInterval, 410
touchgfx::GenericCallback< T1, void, void >	setWindowTitle, 410
-	
~GenericCallback, 362	taskEntry, 411
execute, 362	touchgfx::HAL
isValid, 362	~HAL, 380
touchgfx::GenericCallback< void >	allowDMATransfers, 380
∼GenericCallback, 363	backPorchExited, 380
execute, 363	beginFrame, 380
isValid, 363	blitCopy, 380–382
touchgfx::GestureEvent	blitCopyARGB8888, 382
GestureEvent, 365	blitCopyGlyph, 383
GestureType, 364	blitFill, 384
getEventType, 365	blitSetTransparencyKey, 385
getType, 365	blockCopy, 385
getVelocity, 365	cacheTextString, 385
getX, 366	configureInterrupts, 386
getY, 366	configurePartialFrameBuffer, 386
touchgfx::Gestures	copyFBRegionToMemory, 386, 387
Gestures, 367	disableInterrupts, 387
registerClickEvent, 367	drawDrawableInDynamicBitmap, 387
registerDragEvent, 367	enableInterrupts, 388
registerEventListener, 368	enableLCDControllerInterrupt, 388
setDragThreshold, 368	enableMCULoadCalculation, 388
tick, 368	endFrame, 388
touchgfx::GlyphNode	flushDMA, 388
advance, 369	flushFrameBuffer, 389
height, 369	FrameRefreshStrategy, 379
kerningTablePos, 370	frontPorchEntered, 389
setTop, 370	getAnimationStorage, 389
top, 370	getAuxiliaryLCD, 389
width, 370	getBlitCaps, 390
touchgfx::Gradients	getButtonController, 390
Gradients, 373	getCPUCycles, 390
touchgfx::HALSDL2	getClientFrameBuffer, 390
blitSetTransparencyKey, 404	getDMAType, 391
blockCopy, 404	getDiNAType, 391 getDisplayHeight, 390
• •	
configured Chipterrupt, 404	getDisplayOrientation, 391
configureLCDInterrupt, 404	getDisplayWidth, 391

getFingerSize, 391	setlconXY, 417
getFrameBufferAllocator, 391	setIconX, 417
getFrameRefreshStrategy, 392	setIconY, 417
getInstance, 392	touchgfx::Image
getLCDRefreshCount, 392	draw, 419
getMCULoadPct, 392	getAlpha, 419
getTFTCurrentLine, 392	getBitmap, 419
getTFTFrameBuffer, 393	getSolidRect, 419
getTouchSampleRate, 393	getType, 419
HAL, 379	Image, 418
initialize, 393	setAlpha, 420
lcd, 393	setBitmap, 420
lockDMAToFrontPorch, 394	touchgfx::ImageButtonStyle
lockFrameBuffer, 394	getCurrentlyDisplayedBitmap, 421
noTouch, 394	setBitmapXY, 422
performDisplayOrientationChange, 394	setBitmaps, 422
registerEventListener, 395	touchgfx::ImageProgress
registerTaskDelayFunction, 395	~ImageProgress, 423
registerTextCache, 395	getAlpha, 423
sampleKey, 396	getAnchorAtZero, 424
setAuxiliaryLCD, 396	getBitmap, 424
setButtonController, 396	ImageProgress, 423
setDisplayOrientation, 396 setDragThreshold, 397	setAlpha, 424 setAnchorAtZero, 425
setDrag rifeshold, 397 setFingerSize, 397	setBitmap, 425
set ingersize, 397 setFrameBufferAllocator, 397	setProgressIndicatorPosition, 425
set FrameBufferStartAddress, 398	setValue, 426
set rameBufferStartAddresses, 398	touchgfx::InternalFlashFont
setFrameRateCompensation, 398	getKerning, 427
setFrameRefreshStrategy, 399	getPixelData, 428
setMCUActive, 399	InternalFlashFont, 427
setMCUInstrumentation, 399	touchgfx::JSMOCHelper
setTFTFrameBuffer, 400	draw, 430
setTouchSampleRate, 400	getCachedAbsX, 430
signalDMAInterrupt, 400	getCachedAbsY, 431
swapFrameBuffers, 400	getCachedVisibleRect, 431
taskDelay, 400	getHeight, 431
taskEntry, 401	getWidget, 431
tick, 401	getWidth, 431
touch, 401	JSMOCHelper, 430
unlockFrameBuffer, 401	setWidget, 432
vSync, 401	touchgfx::Keyboard
touchgfx::I2CTouchController	∼Keyboard, 435
\sim I2CTouchController, 414	draw, 435
I2CTouchController, 414	getBuffer, 435
init, 414	getBufferPosition, 436
sampleTouch, 414	getCallbackAreaForCoordinates, 436
touchgfx::I2C	getCharForKey, 436
\sim l2C, 412	getKeyForCoordinates, 436
I2C, 412	getKeyMappingList, 437
init, 412	getLayout, 437
readRegister, 412	getType, 437
writeRegister, 413	handleClickEvent, 437
touchgfx::lconButtonStyle	handleDragEvent, 438
getCurrentlyDisplayedIcon, 416	Keyboard, 435
getIconX, 416	setBuffer, 438
getlconY, 416	setBufferPosition, 438
setIconBitmaps, 416	setKeyListener, 439

setKeymappingList, 439	nextPixel, 482
setLayout, 439	touchgfx::LCD1bpp
setTextIndentation, 439	bitDepth, 484
setupDrawChain, 440	blitCopy, 484, 485
touchgfx::LCD16bpp	blitCopyRLE, 485
bitDepth, 459	copyFrameBufferRegionToMemory, 486
blitCopy, 459, 460	copyRect, 486
blitCopyARGB8888, 461	drawGlyph, 487
blitCopyAlphaPerPixel, 460	drawPartialBitmap, 488
blitCopyL8, 461	drawTextureMapScanLine, 488
blitCopyL8_ARGB8888, 462	fillMemory, 489
blitCopyL8_RGB565, 462	fillRect, 489
blitCopyL8_RGB888, 463	framebufferFormat, 489
copyFrameBufferRegionToMemory, 463	framebufferStride, 490
drawGlyph, 464	getBlueColor, 490
drawPartialBitmap, 464	getBlueFromColor, 490
drawTextureMapScanLine, 465	getColorFrom24BitRGB, 490
fillRect, 466	getColorFromRGB, 491
framebufferFormat, 466	getFramebufferStride, 491
framebufferStride, 466	getGreenColor, 491
getBlueColor, 466	getGreenFromColor, 492
getBlueFromColor, 467	getRedColor, 492
getColorFrom24BitRGB, 467	getRedFromColor, 492
getColorFromRGB, 467	nextLine, 493
getFramebufferStride, 468	nextPixel, 493
getGreenColor, 468	touchgfx::LCD24DebugPrinter
getGreenFromColor, 468	draw, 506
getRedColor, 469	touchgfx::LCD24bpp
getRedFromColor, 469	bitDepth, 496
init, 469	blitCopy, 496
nextLine, 469	blitCopyARGB8888, 497
nextPixel, 470	blitCopyL8, 497
touchgfx::LCD16bppSerialFlash	blitCopyL8_ARGB8888, 498
bitDepth, 473	blitCopyL8 RGB888, 498
blitCopy, 473	copyFrameBufferRegionToMemory, 498
blitCopyARGB8888, 474	drawGlyph, 499
blitCopyL8, 474	drawPartialBitmap, 500
blitCopyL8 ARGB8888, 475	drawTextureMapScanLine, 500
blitCopyL8_RGB565, 475	fillRect, 501
copyFrameBufferRegionToMemory, 475	framebufferFormat, 501
drawGlyph, 476	framebufferStride, 501
drawPartialBitmap, 477	getBlueColor, 502
drawTextureMapScanLine, 477	getBlueFromColor, 502
fillRect, 478	getColorFrom24BitRGB, 502
framebufferFormat, 478	getColorFromRGB, 503
framebufferStride, 478	getFramebufferStride, 503
getBlueColor, 479	getGreenColor, 503
getBlueFromColor, 479	getGreenFromColor, 504
getColorFrom24BitRGB, 479	getRedColor, 504
getColorFromRGB, 480	getRedFromColor, 504
getFramebufferStride, 480	init, 504
getGreenColor, 480	nextLine, 505
getGreenFromColor, 481	nextPixel, 505
getRedColor, 481	touchgfx::LCD2bpp
getRedFromColor, 481	bitDepth, 508
init, 482	blitCopy, 508, 509
LCD16bppSerialFlash, 472	blitCopyAlphaPerPixel, 509
nextLine, 482	copyFrameBufferRegionToMemory, 510
•	.,

copyRect, 510	framebufferStride, 537
drawGlyph, 511	getBlueColor, 537
drawPartialBitmap, 512	getBlueFromColor, 537
drawTextureMapScanLine,	512 getColorFrom24BitRGB, 538
fillRect, 513	getColorFromRGB, 538
framebufferFormat, 513	getFramebufferStride, 539
framebufferStride, 513	getGreenColor, 539
getBlueColor, 514	getGreenFromColor, 539
getBlueFromColor, 514	getRedColor, 539
getColorFrom24BitRGB, 51	
getColorFromRGB, 515	init, 540
getFramebufferStride, 515	nextLine, 540
getGreenColor, 515	nextPixel, 541
getGreenFromColor, 516	touchgfx::LCD8bpp_ABGR2222
_	• • • • •
getRedColor, 516	bitDepth, 543
getRedFromColor, 516	blitCopy, 543, 544
init, 516	blitCopyARGB8888, 544
nextLine, 517	blitCopyAlphaPerPixel, 544
nextPixel, 517	copyFrameBufferRegionToMemory, 545
touchgfx::LCD32bpp	drawGlyph, 545
bitDepth, 519	drawPartialBitmap, 546
blitCopy, 519, 520	drawTextureMapScanLine, 547
blitCopyL8, 520	fillRect, 547
blitCopyL8_ARGB8888, 52	1 framebufferFormat, 548
blitCopyL8_RGB565, 521	framebufferStride, 548
blitCopyL8_RGB888, 522	getBlueColor, 548
blitCopyRGB565, 522	getBlueFromColor, 548
blitCopyRGB888, 523	getColorFrom24BitRGB, 549
copyFrameBufferRegionToI	Memory, 523 getColorFromRGB, 549
drawGlyph, 524	getFramebufferStride, 549
drawPartialBitmap, 524	getGreenColor, 550
drawTextureMapScanLine,	<u> </u>
fillRect, 525	getRedColor, 550
framebufferFormat, 526	getRedFromColor, 551
framebufferStride, 526	init, 551
getBlueColor, 526	nextLine, 551
getBlueFromColor, 527	nextPixel, 551
getColorFrom24BitRGB, 52	
getColorFromRGB, 527	bitDepth, 554
_	blitCopy, 554
getFramebufferStride, 528	· ·
getGreenColor, 528	blitCopyARGB8888, 555
getGreenFromColor, 528	blitCopyAlphaPerPixel, 555
getRedColor, 528	copyFrameBufferRegionToMemory, 556
getRedFromColor, 529	drawGlyph, 556
init, 529	drawPartialBitmap, 557
nextLine, 529	drawTextureMapScanLine, 557
nextPixel, 530	fillRect, 558
touchgfx::LCD4bpp	framebufferFormat, 558
bitDepth, 532	framebufferStride, 559
blitCopy, 532, 533	getBlueColor, 559
blitCopyAlphaPerPixel, 533	getBlueFromColor, 559
copyFrameBufferRegionToI	Memory, 533 getColorFrom24BitRGB, 560
copyRect, 534	getColorFromRGB, 560
drawGlyph, 534	getFramebufferStride, 560
drawPartialBitmap, 535	getGreenColor, 561
drawTextureMapScanLine,	
fillRect, 536	getRedColor, 561
framebufferFormat, 537	getRedFromColor, 562
	9 ,

init ECO	divOEE 446
init, 562 nextLine, 562	div255, 446 div255g, 446
	div255g, 446 div255rb, 447
nextPixel, 562	•
touchgfx::LCD8bpp_BGRA2222	drawChab 447
bitDepth, 565	drawdlyph, 447
blitCopy, 565	drawHorizontalLine, 448
blitCopyARGB8888, 566	drawPartialBitmap, 449
blitCopyAlphaPerPixel, 566	drawRect, 449
copyFrameBufferRegionToMemory, 567	drawString, 449
drawGlyph, 567	drawStringLTR, 450
drawPartialBitmap, 568	drawStringRTL, 450
drawTextureMapScanLine, 568	drawTextureMapScanLine, 451
fillRect, 569	drawTextureMapTriangle, 451
framebufferFormat, 569	drawVerticalLine, 452
framebufferStride, 570	fillRect, 452
getBlueColor, 570	framebufferFormat, 453
getBlueFromColor, 570	framebufferStride, 453
getColorFrom24BitRGB, 571	getBlueColor, 453
getColorFromRGB, 571	getColorFrom24BitRGB, 454
getFramebufferStride, 571	getGreenColor, 454
getGreenColor, 572	getNumLines, 454
getGreenFromColor, 572	getRedColor, 455
getRedColor, 572	init, 455
getRedFromColor, 573	realX, 455
init, 573	realY, 456
nextLine, 573	rotateRect, 456
nextPixel, 573	stringWidth, 457
touchgfx::LCD8bpp_RGBA2222	touchgfx::LED
bitDepth, 576	get, 585
blitCopy, 576	init, 585
blitCopyARGB8888, 577	off, 586
blitCopyAlphaPerPixel, 577	on, 586
copyFrameBufferRegionToMemory, 578	toggle, 586
drawGlyph, 578	touchgfx::Line
drawPartialBitmap, 579	drawCanvasWidget, 589
drawTextureMapScanLine, 579	getEnd, 589
fillRect, 580	getLineEndingStyle, 589
framebufferFormat, 580	getLineWidth, 590
framebufferStride, 581	getMinimalRect, 590
getBlueColor, 581	getStart, 591
getBlueFromColor, 581	LINE_ENDING_STYLE, 588
getColorFrom24BitRGB, 582	Line, 588
getColorFromRGB, 582	setCapPrecision, 591
getFramebufferStride, 582	setEnd, 591, 592
getGreenColor, 583	setLine, 592
getGreenFromColor, 583	setLineEndingStyle, 593
getRedColor, 583	setLineWidth, 593, 594
getRedFromColor, 584	setStart, 594, 595
init, 584	updateEnd, 595, 596
nextLine, 584	updateLengthAndAngle, 596
nextPixel, 584	updateLineWidth, 596, 597
touchgfx::LCD::StringVisuals	updateStart, 597, 598
StringVisuals, 864, 865	touchgfx::LineProgress
touchgfx::LCD	~LineProgress, 599
~LCD, 443	getAlpha, 600
bitDepth, 444	getEnd, 600
blitCopy, 444 copyFrameBufferRegionToMemory, 445	getLineEndingStyle, 600 getLineWidth, 600
Sopyr rame bundin tegion noisie mory, 440	getEnie Width, 000

getStart, 601	concatenateZScale, 614
LineProgress, 599	concatenateZTranslation, 615
setAlpha, 601	getElement, 615
setEnd, 601	identity, 615
setLineEndingStyle, 602	Matrix4x4, 612
setLineWidth, 602	setElement, 615
setPainter, 602	setViewDistance, 616
setProgressIndicatorPosition, 602	touchgfx::ModalWindow
setStart, 603	\sim ModalWindow, 620
setValue, 603	add, 620
touchgfx::ListLayout	getBackgroundHeight, 620
~ListLayout, 605	getBackgroundWidth, 621
add, 605	getShadeAlpha, 621
	•
getDirection, 605	getShadeColor, 621
getType, 605	hide, 621
insert, 605	isShowing, 621
ListLayout, 604	ModalWindow, 620
remove, 606	remove, 621
removeAll, 606	setBackground, 622
setDirection, 606	setShadeAlpha, 622
touchgfx::LockFreeDMA_Queue	setShadeColor, 623
first, 608	show, 623
isEmpty, 608	touchgfx::MoveAnimator
isFull, 608	~MoveAnimator, 625
LockFreeDMA_Queue, 607	cancelMoveAnimation, 625
pop, 608	clearMoveAnimationEndedAction, 625
pushCopyOf, 609	getMoveAnimationDelay, 625
touchgfx::MCUInstrumentation	handleTickEvent, 625
-	
~MCUInstrumentation, 617	isMoveAnimationRunning, 625
getCCConsumed, 617	isRunning, 626
getCPUCycles, 617	MoveAnimator, 624
getElapsedUS, 617	nextMoveAnimationStep, 626
init, 618	setMoveAnimationDelay, 626
MCUInstrumentation, 617	setMoveAnimationEndedAction, 626
setCCConsumed, 618	startMoveAnimation, 626
setMCUActive, 618	touchgfx::NoDMA
touchgfx::MVPApplication	flush, 630
~MVPApplication, 628	getBlitCaps, 630
evaluatePendingScreenTransition, 628	NoDMA, 630
handlePendingScreenTransition, 628	setupDataCopy, 631
MVPApplication, 628	setupDataFill, 631
touchgfx::MVPHeap	signalDMAInterrupt, 631
~MVPHeap, 629	touchgfx::NoTouchController
MVPHeap, 629	~NoTouchController, 632
• •	
touchgfx::ManyBlockAllocator	init, 632
allocateBlock, 610	sampleTouch, 632
freeBlockAfterTransfer, 610	touchgfx::NoTransition
getBlockForTransfer, 610	∼NoTransition, 633
hasBlockReadyForTransfer, 611	handleTickEvent, 633
markBlockReadyForTransfer, 611	NoTransition, 633
touchgfx::Matrix4x4	touchgfx::OSWrappers
concatenateXRotation, 612	giveFrameBufferSemaphore, 634
concatenateXScale, 612	giveFrameBufferSemaphoreFromISR, 634
concatenateXTranslation, 613	initialize, 634
concatenateYRotation, 613	signalVSync, 634
concatenateYScale, 613	takeFrameBufferSemaphore, 635
concatenate Y Translation, 614	taskDelay, 635
concatenate/Rotation, 614	tryTakeFrameBufferSemaphore, 635
oondatonatozi totation, ora	iry ranci ramobalieroemaphore, 000

waitForVSync, 635	setAlpha, 658
touchgfx::Outline	setBitmap, 658
\sim Outline, 637	touchgfx::PainterARGB8888L8Bitmap
getCells, 637	getAlpha, 659
getNumCells, 637	PainterARGB8888L8Bitmap, 659
lineTo, 637	render, 660
moveTo, 638	renderInit, 660
Outline, 637	renderNext, 660
OutlineFlags_t, 637	setAlpha, 661
reset, 638	setBitmap, 661
setMaxRenderY, 638	touchgfx::PainterBGRA2222
wasOutlineTooComplex, 638	getAlpha, 663
touchgfx::PainterABGR2222	getColor, 663
getAlpha, 640	PainterBGRA2222, 662
getColor, 640	render, 663
PainterABGR2222, 640	renderNext, 664
render, 640	setAlpha, 664
renderNext, 641	setColor, 664
setAlpha, 641	touchgfx::PainterBGRA2222Bitmap
setColor, 642	getAlpha, 666
touchgfx::PainterABGR2222Bitmap	PainterBGRA2222Bitmap, 666
getAlpha, 643	render, 666
PainterABGR2222Bitmap, 643	renderlnit, 667
render, 643	renderNext, 667
renderlnit, 644	setAlpha, 667
renderNext, 644	setBitmap, 668
setAlpha, 645	touchgfx::PainterBWBitmap
setBitmap, 645	PainterBWBitmap, 671
touchgfx::PainterARGB2222	render, 672
getAlpha, 646	renderlnit, 672
getColor, 647	renderNext, 672
PainterARGB2222, 646	setBitmap, 673
render, 647	touchgfx::PainterBW
renderNext, 647	bw, 669
setAlpha, 648	getColor, 669
setColor, 648	render, 669 renderNext, 670
touchgfx::PainterARGB2222Bitmap	setColor, 670
getAlpha, 650 PainterARGB2222Bitmap, 649	touchgfx::PainterGRAY2
render, 650	getAlpha, 674
renderl, 650	getColor, 674
rendermit, 650 renderNext, 651	PainterGRAY2, 674
setAlpha, 651	render, 675
setBitmap, 651	renderNext, 675
touchgfx::PainterARGB8888	setAlpha, 675
getAlpha, 653	setColor, 676
getColor, 653	touchgfx::PainterGRAY2Bitmap
PainterARGB8888, 652	getAlpha, 677
render, 653	PainterGRAY2Bitmap, 677
renderNext, 654	render, 678
setAlpha, 654	renderInit, 678
setColor, 654	renderNext, 678
touchgfx::PainterARGB8888Bitmap	setAlpha, 679
getAlpha, 656	setBitmap, 679
PainterARGB8888Bitmap, 656	touchgfx::PainterGRAY4
render, 656	getAlpha, 680
renderlnit, 657	getColor, 681
renderNext, 657	PainterGRAY4, 680

render, 681	setAlpha, 706
renderNext, 681	setBitmap, 706
setAlpha, 682	touchgfx::PainterRGBA2222
setColor, 682	getAlpha, 708
touchgfx::PainterGRAY4Bitmap	getColor, 708
getAlpha, 684	PainterRGBA2222, 708
PainterGRAY4Bitmap, 683	render, 708
render, 684	renderNext, 709
renderInit, 685	setAlpha, 709
renderNext, 685	setColor, 710
setAlpha, 685	touchgfx::PainterRGBA2222Bitmap
setBitmap, 685	getAlpha, 711
touchgfx::PainterRGB565	PainterRGBA2222Bitmap, 711
getAlpha, 687	render, 711
getColor, 687	renderInit, 712
PainterRGB565, 687	renderNext, 712
render, 688	setAlpha, 713
renderNext, 688	setBitmap, 713
setAlpha, 688	touchgfx::Pair
setColor, 689	Pair, 714
touchgfx::PainterRGB565Bitmap	touchgfx::Partition
getAlpha, 690	capacity, 716
PainterRGB565Bitmap, 690	element, 716, 717
render, 691	element_size, 717
renderInit, 691	Partition, 716
renderNext, 691	SupportedTypesList, 716
setAlpha, 692	touchgfx::PixelDataWidget
setBitmap, 692	draw, 719
touchgfx::PainterRGB565L8Bitmap	getAlpha, 719
getAlpha, 694	getSolidRect, 719
PainterRGB565L8Bitmap, 693	PixelDataWidget, 718
render, 694	setAlpha, 719
renderInit, 694	setBitmapFormat, 720
renderNext, 695	setPixelData, 720
setAlpha, 695	touchgfx::Point
setBitmap, 695	dist_sqr, 720
touchgfx::PainterRGB888	touchgfx::Point4
getAlpha, 697	Point4, 722
getColor, 697	touchgfx::PreRenderable
PainterRGB888, 697	draw, 723
render, 697	isPreRendered, 723
renderNext, 698	preRender, 723
setAlpha, 698	PreRenderable, 723
setColor, 700	setupDrawChain, 724
touchgfx::PainterRGB888Bitmap	touchgfx::Presenter
getAlpha, 701	~Presenter, 724
PainterRGB888Bitmap, 701	activate, 725
render, 702	deactivate, 725
renderlnit, 702	Presenter, 725
renderNext, 702	touchgfx::Quadruple
setAlpha, 703	getElement, 736
setBitmap, 703	getW, 736
touchgfx::PainterRGB888L8Bitmap	getX, 737
getAlpha, 705	getY, 737
PainterRGB888L8Bitmap, 704	getZ, 737
render, 705	Quadruple, 736
render, 705	setElement, 737
renderNext, 706	setW, 737
Tondon tord, 700	35(11, 707

setX, 738	getRenderingBuffer, 759
setY, 738	render, 759
setZ, 738	Renderer, 759
touchgfx::RadioButton	setRenderingBuffer, 759
\sim RadioButton, 740	touchgfx::RenderingBuffer
draw, 740	\sim RenderingBuffer, 761
getAlpha, 740	attach, 762
getCurrentlyDisplayedBitmap, 741	getHeight, 762
getDeselectionEnabled, 741	getWidth, 762
getSelected, 741	getXAdjust, 762
getSolidRect, 741	inbox, 763
getType, 741	RenderingBuffer, 761
handleClickEvent, 742	row, 763
RadioButton, 740	touchgfx::RepeatButton
setAlpha, 742	getDelay, 764
setBitmaps, 742	getInterval, 765
setDeselectedAction, 743	handleClickEvent, 765
setDeselectionEnabled, 743	handleTickEvent, 765
setSelected, 743	RepeatButton, 764
touchgfx::RadioButtonGroup	setDelay, 765
\sim RadioButtonGroup, 745	setInterval, 766
add, 745	touchgfx::RepeatButtonTrigger
getDeselectionEnabled, 745	getDelay, 767
getRadioButton, 746	getInterval, 767
getSelectedRadioButton, 746	handleClickEvent, 767
getSelectedRadioButtonIndex, 746	setDelay, 767
radioButtonClickedHandler, 746	setInterval, 768
radioButtonDeselectedHandler, 747	touchgfx::SDL2TouchController
RadioButtonGroup, 745	init, 828
setDeselectionEnabled, 747	sampleTouch, 828
setRadioButtonDeselectedHandler, 747	touchgfx::SDLTouchController
setRadioButtonSelectedHandler, 747	init, 829
setSelected, 749	sampleTouch, 829
touchgfx::Rasterizer	touchgfx::ScalableImage
calculateAlpha, 751	\sim ScalableImage, 769
FillingRule, 751	draw, 770
lineTo, 751	drawTriangle, 770
moveTo, 752	getAlpha, 770
Rasterizer, 751	getBitmap, 771
render, 752	getScalingAlgorithm, 771
reset, 752	getSolidRect, 771
setFillingRule, 752	getType, 771
setMaxRenderY, 753	lookupRenderVariant, 772
wasOutlineTooComplex, 753	ScalableImage, 769
touchgfx::Rect	ScalingAlgorithm, 769
area, 755	setAlpha, 772
bottom, 755	setBitmap, 772
expandToFit, 755	setScalingAlgorithm, 772
includes, 756	touchgfx::Scanline
intersect, 756	∼Scanline, 774
isEmpty, 756	addCell, 774
operator!=, 757	addSpan, 776
operator==, 758	getNumSpans, 776
operator&, 757	getY, 776
operator&=, 757	isReady, 776
Rect, 754	reset, 777
right, 758	resetSpans, 777
touchgfx::Renderer	Scanline, 774

touchgfx::Scanline::iterator	setEasingEquation, 806
getCovers, 429	setHeight, 806
getNumPix, 429	setHorizontal, 806
iterator, 428	setItemPressedCallback, 807
next, 429	setItemSelectedCallback, 807
touchgfx::Screen	setMaxSwipeItems, 807
\sim Screen, 778	setNumberOfItems, 808
add, 779	setOffset, 808
afterTransition, 779	setSwipeAcceleration, 808
bindTransition, 779	setWidth, 809
draw, 779, 780	stopAnimation, 809
getRootContainer, 780	touchgfx::ScrollList
handleClickEvent, 780	\sim ScrollList, 810
handleDragEvent, 780	getItem, 811
handleGestureEvent, 781	getNearestAlignedOffset, 811
handleKeyEvent, 781	getPaddingAfter, 811
handleTickEvent, 781	getPaddingBefore, 811
JSMOC, 781	getPositionForItem, 812
remove, 781	getSnapping, 812
Screen, 778	handleClickEvent, 812
setupScreen, 782	keepOffsetInsideLimits, 813
startSMOC, 782	ScrollList, 810
tearDownScreen, 782	setDrawables, 813
useSMOCDrawing, 782	setPadding, 813
usingSMOC, 783	setSnapping, 814
touchgfx::ScrollBase	setWindowSize, 814
~ScrollBase, 797	touchgfx::ScrollWheel
allowHorizontalDrag, 798	\sim ScrollWheel, 815
allowVerticalDrag, 798	ScrollWheel, 815
animateToltem, 798	setDrawables, 815
animateToPosition, 799	touchgfx::ScrollWheelBase
AnimationState, 797	\sim ScrollWheelBase, 817
getAnimationSteps, 799	animateToPosition, 817
getCircular, 799	getPositionForItem, 817
getDragAcceleration, 799	getSelectedItem, 817
getDrawableMargin, 800	getSelectedItemOffset, 818
getDrawableSize, 800	handleClickEvent, 818
getHorizontal, 800	handleDragEvent, 818
getMaxSwipeItems, 800	handleGestureEvent, 818
getNearestAlignedOffset, 801	keepOffsetInsideLimits, 819
getNormalizedOffset, 801	ScrollWheelBase, 817
getNumberOfItems, 801	setAnimateToCallback, 819
getOffset, 802	setSelectedItemOffset, 819
getPositionForItem, 802	touchgfx::ScrollWheelWithSelectionStyle
getSwipeAcceleration, 802	~ScrollWheelWithSelectionStyle, 822
handleDragEvent, 803	getSelectedItemExtraSizeAfter, 822
handleGestureEvent, 803	· ·
,	getSelectedItemExtraSizeBefore, 822
handleTickEvent, 803	getSelectedItemMarginAfter, 822
initialize, 803	getSelectedItemMarginBefore, 823
isAnimating, 803	initialize, 823
itemChanged, 804	itemChanged, 823
keepOffsetInsideLimits, 804	refreshDrawableListsLayout, 823
ScrollBase, 797	ScrollWheelWithSelectionStyle, 821
setAnimationEndedCallback, 804	setCircular, 823
setAnimationSteps, 805	setDrawableSize, 824
setCircular, 805	setDrawables, 824
setDragAcceleration, 805	setHeight, 825
setDrawableSize, 806	setHorizontal, 825

setNumberOfItems, 825	animationEndedHandler, 839
setOffset, 826	getAnimationDuration, 839
setSelectedItemExtraSize, 826	getAnimationEasingEquation, 840
setSelectedItemMargin, 826	getBackgroundX, 840
setSelectedItemOffset, 827	getBackgroundY, 840
setSelectedItemPosition, 827	getCollapsedXCoordinate, 840
setWidth, 827	getCollapsedYCoordinate, 840
touchgfx::ScrollableContainer	getExpandDirection, 840
~ScrollableContainer, 786	getExpandedStateTimeout, 841
add, 786	getExpandedStateTimer, 841
childGeometryChanged, 787	getExpandedXCoordinate, 841
doScroll, 787	getExpandedYCoordinate, 841
enableHorizontalScroll, 787	getHiddenPixelsWhenExpanded, 841
enableVerticalScroll, 788	getState, 841
getContainedArea, 788	getStateChangeButtonX, 842
getLastChild, 788	getStateChangeButtonY, 842
getScrolledX, 788	getVisiblePixelsWhenCollapsed, 842
getScrolledY, 789	remove, 842
getType, 789	resetExpandedStateTimer, 842
getXBorder, 789	setAnimationDuration, 843
getXScrollbar, 789	setAnimationEasingEquation, 843
getYBorder, 790	setExpandDirection, 843
getYScrollbar, 790	setExpandedStateTimeout, 843
handleClickEvent, 790	setHiddenPixelsWhenExpanded, 843
handleDragEvent, 790	setState, 844
handleGestureEvent, 791	setStateChangedAnimationEndedCallback, 844
handleTickEvent, 791	setStateChangedCallback, 844
invalidateScrollbars, 791	setVisiblePixelsWhenCollapsed, 845
isScrollableXY, 791	setup, 844, 845
moveChildrenRelative, 791	stateChangeButtonClickedHandler, 845
reset, 792	touchgfx::SlideTransition
ScrollableContainer, 786	~SlideTransition, 857
setMaxVelocity, 792	handleTickEvent, 857
setScrollThreshold, 793	
setScrollbarPadding, 793	init, 857
setScrollbarWidth, 793	initMoveDrawable, 857 SlideTransition, 857
setScrollbarsAlpha, 792 setScrollbarsColor, 793	tearDown, 858 tickMoveDrawable, 858
setScrollbarsPermanentlyVisible, 793	touchgfx::Slider
	getIndicatorMax, 848
setScrollbarsVisible, 793	
touchgfx::Shape ~Shape, 831	getIndicatorMin, 848 getIndicatorPositionRangeSize, 849
getCacheX, 831	getIndicatorPositionHangeSize, 849 getIndicatorRadius, 849
•	_
getCacheY, 831	getMaxValue, 849
getCornerV, 832	getMinValue, 849
getCornerY, 832	getType, 850
getNumPoints, 832	getValue, 850
setCarner 832	getValueRangeSize, 850
setCorner, 833	handleClickEvent, 850
touchgfx::SingleBlockAllocator	handleDragEvent, 851
allocateBlock, 834	positionToValue, 851
freeBlockAfterTransfer, 835	setBitmaps, 851, 852
getBlockForTransfer, 835	setNewValueCallback, 852
hasBlockReadyForTransfer, 835	setStartValueCallback, 852
markBlockReadyForTransfer, 836	setStopValueCallback, 852
touchgfx::SlideMenu	setValue, 854
ann XXV	
add, 839 animateToState, 839	setValueRange, 854, 855 setupHorizontalSlider, 853

setupVerticalSlider, 853	setWideTextAction, 877
Slider, 848	setXBaselineY, 877
updateIndicatorPosition, 855	TextArea, 871
valueToPosition, 855	touchgfx::TextAreaWithOneWildcard
touchgfx::Snapper	draw, 879
\sim Snapper, 859	getTextHeight, 879
handleClickEvent, 859	getTextWidth, 879
handleDragEvent, 860	getType, 880
setDragAction, 860	getWildcard, 880
setSnapPosition, 860	setWildcard, 880
setSnappedAction, 860	TextAreaWithOneWildcard, 879
Snapper, 859	touchgfx::TextAreaWithTwoWildcards
touchgfx::SnapshotWidget	draw, 881
\sim SnapshotWidget, 862	getTextHeight, 882
draw, 862	getTextWidth, 882
getAlpha, 862	getType, 882
getSolidRect, 863	getWildcard1, 882
getType, 863	getWildcard2, 883
makeSnapshot, 863	setWildcard1, 883
setAlpha, 863	setWildcard2, 883
SnapshotWidget, 862	TextAreaWithTwoWildcards, 881
touchgfx::SwipeContainer	touchgfx::TextAreaWithWildcardBase
add, 866	calculateTextHeight, 884
getNumberOfPages, 866	TextAreaWithWildcardBase, 884
handleClickEvent, 867	touchgfx::TextButtonStyle
handleDragEvent, 867	setText, 886
handleGestureEvent, 867	setTextColors, 886
handleTickEvent, 867	setTextOolors, 886
remove, 868	setTextRotation, 887
setEndSwipeElasticWidth, 868	setTextXY, 887
setPageIndicatorBitmaps, 868	setTextX, 887
setPageIndicatorXYWithCenteredX, 869	setTextX, 887
setPageIndicatorXY, 868	touchgfx::TextProgress
_	
setSelectedPage, 869	~TextProgress, 889
setSwipeCutoff, 869	getAlpha, 889
touchgfx::TextArea	getColor, 889
draw, 872	getNumberOfDecimals, 889
getAlpha, 872	getTypedText, 890
getColor, 872	setAlpha, 890
getIndentation, 872	setColor, 890
getLinespacing, 872	setNumberOfDecimals, 891
getRotation, 873	setProgressIndicatorPosition, 891
getSolidRect, 873	setTypedText, 891
getTextHeight, 873	setValue, 892
getTextWidth, 873	TextProgress, 889
getType, 874	touchgfx::TextProvider
getTypedText, 874	getNextChar, 893
getWideTextAction, 874	getNextLigature, 893, 894
resizeHeightToCurrentText, 874	initialize, 895
resizeToCurrentText, 875	TextProvider, 893
resizeToCurrentTextWithAlignment, 875	touchgfx::Texts
setAlpha, 875	getLanguage, 896
setBaselineY, 875	getText, 896
setColor, 876	setLanguage, 896
setIndentation, 876	setTranslation, 897
setLinespacing, 876	touchgfx::TextureMapper
setRotation, 877	∼TextureMapper, 901
setTypedText, 877	applyTransformation, 901
setTypedText, 877	applyTransformation, 901

draw, 901	touchgfx::TiledImageButtonStyle
drawTriangle, 902	setHeight, 918
getAlpha, 902	setTileBitmaps, 918
getBitmap, 903	setTileOffset, 919
getBitmapPositionX, 903	setWidth, 919
getBitmapPositionY, 903	touchgfx::ToggleButton
getBoundingRect, 903	forceState, 920
getCameraDistance, 903	getState, 920
getCameraX, 904	getState, 920 getType, 921
getCameraY, 904	handleClickEvent, 921
getOrigoX, 904	setBitmaps, 921
getOrigoY, 904	ToggleButton, 920
getOrigoZ, 904	touchgfx::ToggleButtonTrigger
getRenderingAlgorithm, 905	forceState, 922
getScale, 905	getToggleCanceled, 924
getSolidRect, 905	handleClickEvent, 924
getType, 905	setToggleCanceled, 924
getX0, 905	touchgfx::TouchArea
getX1, 906	draw, 925
getX2, 906	getSolidRect, 925
getX3, 906	getType, 926
getXAngle, 906	handleClickEvent, 926
getY0, 906	handleDragEvent, 926
getY1, 907	setPressedAction, 926
getY2, 907	TouchArea, 925
getY3, 907	touchgfx::TouchButtonTrigger
-	handleClickEvent, 927
getYAngle, 907	
getZ0, 907	touchgfx::TouchCalibration
getZ1, 908	setCalibrationMatrix, 928
getZ2, 908	translatePoint, 928
getZ3, 908	touchgfx::TouchController
getZAngle, 908	\sim TouchController, 929
lookupRenderVariant, 908	init, 929
RenderingAlgorithm, 901	sampleTouch, 929
setAlpha, 909	touchgfx::Transition
setBitmap, 909	\sim Transition, 930
setBitmapPosition, 909	handleTickEvent, 931
setCamera, 910	init, 931
setCameraDistance, 910	isDone, 931
setOrigo, 910, 911	setScreenContainer, 931
setRenderingAlgorithm, 911	tearDown, 931
setScale, 911	Transition, 930
updateAngles, 911	touchgfx::TwoWildcardTextButtonStyle
updateXAngle, 912	setTwoWildcardText, 933
updateYAngle, 912	setTwoWildcardTextColors, 933
updateZAngle, 912	setTwoWildcardTextPosition, 934
touchgfx::TiledImage	setTwoWildcardTextRotation, 934
draw, 914	setTwoWildcardTextXY, 934
getOffset, 914	setTwoWildcardTextX, 934
getSolidRect, 915	setTwoWildcardTextY, 935
getType, 915	setWildcardTextBuffer1, 935
getXOffset, 915	setWildcardTextBuffer2, 935
getYOffset, 915	touchgfx::TypedText
setBitmap, 916	getAlignment, 937
setOffset, 916	getFont, 937
setXOffset, 916	getFontId, 937
setYOffset, 917	getId, 937
TiledImage, 914	getText, 937
	go, oo

getTextDirection, 938	setWildcardTextXY, 965
hasValidId, 938	setWildcardTextX, 964
registerTexts, 938	setWildcardTextY, 965
registerTypedTextDatabase, 938	touchgfx::ZoomAnimationImage
TypedText, 936	\sim ZoomAnimationImage, 969
touchgfx::UIEventListener	getAlpha, 969
\sim UIEventListener, 940	getAnimationDelay, 969
handleClickEvent, 940	getLargeBitmap, 969
handleDragEvent, 940	getScalingMode, 969
handleGestureEvent, 940	getSmallBitmap, 970
handleKeyEvent, 941	getType, 970
handlePendingScreenTransition, 941	handleTickEvent, 970
handleTickEvent, 941	isRunning, 970
touchgfx::Unicode	isZoomAnimationRunning, 970
atoi, 943	setAlpha, 971
fromUTF8, 943	setAnimationDelay, 971
itoa, 943	setAnimationEndedCallback, 971
snprintf, 944	setBitmaps, 971
snprintfFloat, 945, 946	setCurrentState, 972
snprintfFloats, 947, 948	setDimension, 972
strlen, 949, 950	setHeight, 972
	setPosition, 973
strnomp, 950	setScalingMode, 973
strncmp_ignore_white_spaces, 951	setWidth, 973
strncpy, 951	startTimerAndSetParameters, 973
toUTF8, 952	startZoomAndMoveAnimation, 974
UnicodeChar, 942	startZoomAnimation, 975
utoa, 952	States, 968
vsnprintf, 953	updateRenderingMethod, 975
touchgfx::Vector	updateZoomAnimationDeltaXY, 975
add, 955	ZoomAnimationImage, 969
clear, 955	touchgfx::colortype
contains, 955	colortype, 250
isEmpty, 955	getColor32, 250
maxCapacity, 956	operator uint16_t, 250
operator[], 956	touchgfx_generic_init
quickRemoveAt, 956	touchgfx, 67
remove, 957	touchgfx_init
removeAt, 957	touchgfx, 68
reverse, 957	transformDisplayToFrameBuffer
size, 957	touchgfx::DisplayTransformation, 277–279
Vector, 955	transformFrameBufferToDisplay
touchgfx::Vector4	touchgfx::DisplayTransformation, 279
crossProduct, 959	Transition, 930
Vector4, 958	touchgfx::Transition, 930
touchgfx::View	translatePoint
bind, 960	
View, 960	touchgfx::TouchCalibration, 928 translateRectToAbsolute
touchgfx::Widget	
\sim Widget, 961	touchgfx::Drawable, 306
getLastChild, 961	tryTakeFrameBufferSemaphore
getType, 961	touchgfx::OSWrappers, 635
Widget, 961	TwoWildcardTextButtonStyle < T >, 932
touchgfx::WildcardTextButtonStyle	TypedText, 935
setWildcardText, 963	touchgfx::TypedText, 936
setWildcardTextBuffer, 963	TypedText::TypedTextData, 939
setWildcardTextColors, 964	UIEventListener, 939
setWildcardTextPosition, 964	Unicode, 941
setWildcardTextRotation, 964	UnicodeChar
Section and telephone in the section of the section	June 4

touchgfx::Unicode, 942	touchgfx::Slider, 855
unlink	Vector
touchgfx::Container, 259	touchgfx::Vector, 955
unlockFrameBuffer	Vector< T, capacity >, 954
touchgfx::HAL, 401	Vector4, 958
unregisterTimerWidget	touchgfx::Vector4, 958
touchgfx::Application, 156	View
updateAbstractShapeCache	touchgfx::View, 960
touchgfx::AbstractShape, 129	View< T >, 959
updateAngle	vsnprintf
touchgfx::AbstractShape, 129	touchgfx::Unicode, 953
updateAngles	
touchgfx::TextureMapper, 911	waitFlashReadComplete
updateArc	touchgfx::FlashDataReader, 344
touchgfx::Circle, 233	waitForFrameBufferSemaphore
updateArcEnd	touchgfx::DMA_Interface, 285
touchgfx::Circle, 233	waitForVSync
updateArcStart	touchgfx::OSWrappers, 635
touchgfx::Circle, 235	wasOutlineTooComplex
updateCache	touchgfx::Outline, 638
touchgfx::CacheableContainer, 199	touchgfx::Rasterizer, 753
updateClock	WideTextAction
touchgfx::AbstractClock, 75	touchgfx, 57
touchgfx::AnalogClock, 138	Widget, 960
touchgfx::DigitalClock, 277	touchgfx::Widget, 961
updateEnd	width
touchgfx::Line, 595, 596	touchgfx::GlyphNode, 370
updateIndicatorPosition	WildcardTextButtonStyle < T >, 962
touchgfx::Slider, 855	writeRegister
updateLengthAndAngle	touchgfx::I2C, 413
touchgfx::Line, 596	
updateLineWidth	ZoomAnimationImage, 965
touchgfx::Line, 596, 597	touchgfx::ZoomAnimationImage, 969
updateRenderingMethod	
touchgfx::ZoomAnimationImage, 975	
updateScale	
touchgfx::AbstractShape, 130	
updateStart	
touchgfx::Line, 597, 598	
updateTextPosition	
touchgfx::ButtonWithLabel, 196	
updateXAngle	
touchgfx::TextureMapper, 912	
updateYAngle	
touchgfx::TextureMapper, 912	
updateZAngle	
touchgfx::TextureMapper, 912	
updateZoomAnimationDeltaXY	
touchgfx::ZoomAnimationImage, 975	
useSMOCDrawing	
touchgfx::Screen, 782	
usingSMOC	
touchgfx::Screen, 783	
utoa	
touchgfx::Unicode, 952	
0	
vSync	
touchgfx::HAL, 401	
valueToPosition	