

Qwt User's Guide Reference Manual

5.1.2

Generated by Doxygen 1.5.0

Sun Mar 22 16:54:06 2009

Contents

1	Qwt - Qt Widgets for Technical Applications	1
2	Qwt User's Guide Hierarchical Index	3
3	Qwt User's Guide Class Index	6
4	Qwt User's Guide File Index	9
5	Qwt User's Guide Page Index	14
6	Qwt User's Guide Class Documentation	15
7	Qwt User's Guide File Documentation	445
8	Qwt User's Guide Page Documentation	447

1 Qwt - Qt Widgets for Technical Applications

The Qwt library contains GUI Components and utility classes which are primarily useful for programs with a technical background. Beside a 2D plot widget it provides scales, sliders, dials, compasses, thermometers, wheels and knobs to control or display values, arrays, or ranges of type double.

1.1 License

Qwt is distributed under the terms of the [Qwt License, Version 1.0](#).

1.2 Platforms

Qwt 5.x might be usable in all environments where you find [Qt](#). It is compatible with Qt 3.3.x and Qt 4.x, but the documentation is generated for Qt 4.x.

1.3 Screenshots

- [Curve Plots](#)
- [Scatter Plot](#)
- [Spectrogram, Contour Plot](#)
- [Histogram](#)
- [Dials, Compasses, Knobs, Wheels, Sliders, Thermos](#)

Screenshots are only available in the HTML docs.

1.4 Downloads

Stable releases, prereleases and snapshots are available at the [Qwt project page](#).

Qwt doesn't distribute binary packages, but below is a incomplete list of packagers : [Debian](#), [S.u.S.E](#), [Fedora](#), [Gentoo](#), [Fink](#), [Ubuntu](#).

For getting a 5.2 development snapshot from the SVN repository:

```
svn co https://qwt.svn.sourceforge.net/svnroot/qwt/trunk/qwt
```

For getting a snapshot with all bugfixes for the latest 5.1 release:

```
svn co https://qwt.svn.sourceforge.net/svnroot/qwt/branches/qwt-5.1
```

1.5 Installation

Have a look at the qwt.pro project file. It is prepared for building dynamic libraries in Win32 and Unix/X11 environments. If you don't know what to do with it, read the file [INSTALL](#) and/or Trolltechs [qmake](#) documentation. Once you have build the library you have to install all files from the lib, include and doc directories.

1.6 Support

- Mailing list

For all kind of Qwt related questions use the [Qwt mailing list](#).

If you prefer newsgroups use the mail to news gateway of [Gmane](#).

- Forum

[Qt Centre](#) is a great resource for Qt related questions. It has a sub forum, that is dedicated to Qwt related questions.

- Individual support

If you are looking for individual support, or need someone who implements your Qwt component/application contact qwt-support@tigertal.de.

1.7 Related Projects

[QwtPolar](#), a polar plot widget.

[QwtPlot3D](#), an OpenGL 3D plot widget.

[QtiPlot](#), data analysis and scientific plotting tool, using [QwtPlot](#).

1.8 Language Bindings

[PyQwt](#), a set of Qwt Python bindings.

[Korundum/QtRuby](#), including a set of Qwt Ruby bindings.

1.9 Donations

Sourceforge offers a [Donation System](#) via PayPal. You can use it, if you like to [support](#) the development of Qwt.

1.10 Credits:

Authors:

Uwe Rathmann, Josef Wilgen (<= Qwt 0.2)

Project admin:

Uwe Rathmann <rathmann@users.sourceforge.net>

2 Qwt User's Guide Hierarchical Index

2.1 Qwt User's Guide Class Hierarchy

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

QwtAbstractScale	15
QwtKnob	130
QwtSlider	394
QwtThermo	429
QwtAbstractScaleDraw	20
QwtRoundScaleDraw	348
QwtDialScaleDraw	97
QwtScaleDraw	359
QwtArrowButton	48
QwtClipper	51
QwtColorMap	51
QwtAlphaColorMap	38
QwtLinearColorMap	149
QwtCompassRose	63
QwtSimpleCompassRose	390
QwtCurveFitter	77
QwtSplineCurveFitter	406

QwtData	77
QwtArrayData	45
QwtCPointerData	74
QwtPolygonFData	341
QwtDialNeedle	95
QwtCompassMagnetNeedle	59
QwtCompassWindArrow	64
QwtDialSimpleNeedle	99
QwtDoubleInterval	102
QwtDoubleRange	109
QwtAbstractSlider	28
QwtDial	80
QwtAnalogClock	40
QwtCompass	55
QwtKnob	130
QwtSlider	394
QwtWheel	439
QwtCounter	67
QwtDynGridLayout	116
QwtEventPattern	121
QwtPicker	181
QwtPlotPicker	296
QwtPlotZoomer	332
QwtEventPattern::KeyPattern	129
QwtEventPattern::MousePattern	129
QwtIntervalData	129
QwtLegend	135
QwtLegendItemManager	148
QwtPlotItem	265

QwtPlotCurve	239
QwtPlotGrid	258
QwtPlotMarker	287
QwtPlotRasterItem	307
QwtPlotSpectrogram	318
QwtPlotScaleItem	312
QwtPlotSvgItem	328
QwtMagnifier	158
QwtPlotMagnifier	284
QwtMetricsMap	168
QwtPainter	170
QwtPanner	175
QwtPlotPanner	293
QwtPickerMachine	204
QwtPickerClickPointMachine	200
QwtPickerClickRectMachine	201
QwtPickerDragPointMachine	202
QwtPickerDragRectMachine	203
QwtPickerPolygonMachine	206
QwtPlotCanvas	234
QwtPlotDict	256
QwtPlot	209
QwtPlotLayout	276
QwtPlotPrintFilter	304
QwtRasterData	343
QwtRect	345
QwtScaleArithmetic	353
QwtScaleDiv	355
QwtScaleEngine	369

QwtLinearScaleEngine	153
QwtLog10ScaleEngine	156
QwtScaleMap	375
QwtScaleTransformation	379
QwtScaleWidget	381
QwtSpline	403
QwtSymbol	406
QwtText	411
QwtTextEngine	423
QwtMathMLTextEngine	165
QwtPlainTextEngine	207
QwtRichTextEngine	346
QwtTextLabel	425
QwtLegendItem	140

3 Qwt User's Guide Class Index

3.1 Qwt User's Guide Class List

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

QwtAbstractScale (An abstract base class for classes containing a scale)	15
QwtAbstractScaleDraw (A abstract base class for drawing scales)	20
QwtAbstractSlider (An abstract base class for slider widgets)	28
QwtAlphaColorMap (QwtAlphaColorMap variies the alpha value of a color)	38
QwtAnalogClock (An analog clock)	40
QwtArrayData (Data class containing two QwtArray<double> objects)	45
QwtArrowButton (Arrow Button)	48
QwtClipper (Some clipping algos)	51
QwtColorMap (QwtColorMap is used to map values into colors)	51
QwtCompass (A Compass Widget)	55
QwtCompassMagnetNeedle (A magnet needle for compass widgets)	59

QwtCompassRose (Abstract base class for a compass rose)	63
QwtCompassWindArrow (An indicator for the wind direction)	64
QwtCounter (The Counter Widget)	67
QwtCPointerData (Data class containing two pointers to memory blocks of doubles)	74
QwtCurveFitter (Abstract base class for a curve fitter)	77
QwtData (QwtData defines an interface to any type of curve data)	77
QwtDial (QwtDial class provides a rounded range control)	80
QwtDialNeedle (Base class for needles that can be used in a QwtDial)	95
QwtDialScaleDraw (A special scale draw made for QwtDial)	97
QwtDialSimpleNeedle (A needle for dial widgets)	99
QwtDoubleInterval (A class representing an interval)	102
QwtDoubleRange (A class which controls a value within an interval)	109
QwtDynGridLayout (Lays out widgets in a grid, adjusting the number of columns and rows to the current size)	116
QwtEventPattern (A collection of event patterns)	121
QwtEventPattern::KeyPattern (A pattern for key events)	129
QwtEventPattern::MousePattern (A pattern for mouse events)	129
QwtIntervalData (Interval data class)	129
QwtKnob (The Knob Widget)	130
QwtLegend (The legend widget)	135
QwtLegendItem (A legend label)	140
QwtLegendItemManager	148
QwtLinearColorMap (QwtLinearColorMap builds a color map from color stops)	149
QwtLinearScaleEngine (A scale engine for linear scales)	153
QwtLog10ScaleEngine (A scale engine for logarithmic (base 10) scales)	156
QwtMagnifier (QwtMagnifier provides zooming, by magnifying in steps)	158
QwtMathMLTextEngine (Text Engine for the MathML renderer of the Qt solutions package)	165
QwtMetricsMap (A Map to translate between layout, screen and paint device metrics)	168
QwtPainter (A collection of QPainter workarounds)	170

QwtPanner (QwtPanner provides panning of a widget)	175
QwtPicker (QwtPicker provides selections on a widget)	181
QwtPickerClickPointMachine (A state machine for point selections)	200
QwtPickerClickRectMachine (A state machine for rectangle selections)	201
QwtPickerDragPointMachine (A state machine for point selections)	202
QwtPickerDragRectMachine (A state machine for rectangle selections)	203
QwtPickerMachine (A state machine for QwtPicker selections)	204
QwtPickerPolygonMachine (A state machine for polygon selections)	206
QwtPlainTextEngine (A text engine for plain texts)	207
QwtPlot (A 2-D plotting widget)	209
QwtPlotCanvas	234
QwtPlotCurve (A class which draws curves)	239
QwtPlotDict (A dictionary for plot items)	256
QwtPlotGrid (A class which draws a coordinate grid)	258
QwtPlotItem (Base class for items on the plot canvas)	265
QwtPlotLayout (Layout class for QwtPlot)	276
QwtPlotMagnifier (QwtPlotMagnifier provides zooming, by magnifying in steps)	284
QwtPlotMarker (A class for drawing markers)	287
QwtPlotPanner (QwtPlotPanner provides panning of a plot canvas)	293
QwtPlotPicker (QwtPlotPicker provides selections on a plot canvas)	296
QwtPlotPrintFilter (A base class for plot print filters)	304
QwtPlotRasterItem (A class, which displays raster data)	307
QwtPlotScaleItem (A class which draws a scale inside the plot canvas)	312
QwtPlotSpectrogram (A plot item, which displays a spectrogram)	318
QwtPlotSvgItem (A plot item, which displays data in Scalable Vector Graphics (SVG) format)	328
QwtPlotZoomer (QwtPlotZoomer provides stacked zooming for a plot widget)	332
QwtPolygonFData (Data class containing a single <code>QwtArray<QwtDoublePoint></code> object)	341
QwtRasterData (QwtRasterData defines an interface to any type of raster data)	343
QwtRect	345

QwtRichTextEngine (A text engine for Qt rich texts)	346
QwtRoundScaleDraw (A class for drawing round scales)	348
QwtScaleArithmetic (Arithmetic including a tolerance)	353
QwtScaleDiv (A class representing a scale division)	355
QwtScaleDraw (A class for drawing scales)	359
QwtScaleEngine (Base class for scale engines)	369
QwtScaleMap (A scale map)	375
QwtScaleTransformation (Operations for linear or logarithmic (base 10) transformations)	379
QwtScaleWidget (A Widget which contains a scale)	381
QwtSimpleCompassRose (A simple rose for QwtCompass)	390
QwtSlider (The Slider Widget)	394
QwtSpline (A class for spline interpolation)	403
QwtSplineCurveFitter (A curve fitter using cubic splines)	406
QwtSymbol (A class for drawing symbols)	406
QwtText (A class representing a text)	411
QwtTextEngine (Abstract base class for rendering text strings)	423
QwtTextLabel (A Widget which displays a QwtText)	425
QwtThermo (The Thermometer Widget)	429
QwtWheel (The Wheel Widget)	439

4 Qwt User's Guide File Index

4.1 Qwt User's Guide File List

Here is a list of all documented files with brief descriptions:

qwt_abstract_scale.cpp	??
qwt_abstract_scale.h	??
qwt_abstract_scale_draw.cpp	??
qwt_abstract_scale_draw.h	??
qwt_abstract_slider.cpp	??
qwt_abstract_slider.h	??

qwt_analog_clock.cpp	??
qwt_analog_clock.h	??
qwt_array.h	??
qwt_arrow_button.cpp	??
qwt_arrow_button.h	??
qwt_clipper.cpp	??
qwt_clipper.h	??
qwt_color_map.cpp	??
qwt_color_map.h	??
qwt_compass.cpp	??
qwt_compass.h	??
qwt_compass_rose.cpp	??
qwt_compass_rose.h	??
qwt_counter.cpp	??
qwt_counter.h	??
qwt_curve_fitter.cpp	??
qwt_curve_fitter.h	??
qwt_data.cpp	??
qwt_data.h	??
qwt_dial.cpp	??
qwt_dial.h	??
qwt_dial_needle.cpp	??
qwt_dial_needle.h	??
qwt_double_interval.cpp	??
qwt_double_interval.h	??
qwt_double_range.cpp	??
qwt_double_range.h	??
qwt_double_rect.cpp	??
qwt_double_rect.h	445

qwt_dyngrid_layout.cpp	??
qwt_dyngrid_layout.h	??
qwt_event_pattern.cpp	??
qwt_event_pattern.h	??
qwt_global.h	??
qwt_interval_data.cpp	??
qwt_interval_data.h	??
qwt_knob.cpp	??
qwt_knob.h	??
qwt_layout_metrics.cpp	??
qwt_layout_metrics.h	??
qwt_legend.cpp	??
qwt_legend.h	??
qwt_legend_item.cpp	??
qwt_legend_item.h	??
qwt_legend_itemmanager.h	??
qwt_magnifier.cpp	??
qwt_magnifier.h	??
qwt_math.cpp	??
qwt_math.h	??
qwt_mathml_text_engine.cpp	??
qwt_mathml_text_engine.h	??
qwt_paint_buffer.cpp	??
qwt_paint_buffer.h	??
qwtPainter.cpp	??
qwtPainter.h	??
qwt_panner.cpp	??
qwt_panner.h	??
qwt_picker.cpp	??

qwt_picker.h	??
qwt_picker_machine.cpp	??
qwt_picker_machine.h	??
qwt_plot.cpp	??
qwt_plot.h	??
qwt_plot_axis.cpp	??
qwt_plot_canvas.cpp	??
qwt_plot_canvas.h	??
qwt_plot_curve.cpp	??
qwt_plot_curve.h	??
qwt_plot_dict.cpp	??
qwt_plot_dict.h	446
qwt_plot_grid.cpp	??
qwt_plot_grid.h	??
qwt_plot_item.cpp	??
qwt_plot_item.h	??
qwt_plot_layout.cpp	??
qwt_plot_layout.h	??
qwt_plot_magnifier.cpp	??
qwt_plot_magnifier.h	??
qwt_plot_marker.cpp	??
qwt_plot_marker.h	??
qwt_plot_panner.cpp	??
qwt_plot_panner.h	??
qwt_plot_picker.cpp	??
qwt_plot_picker.h	??
qwt_plot_print.cpp	??
qwt_plot_printfilter.cpp	??
qwt_plot_printfilter.h	??

qwt_plot_rasteritem.cpp	??
qwt_plot_rasteritem.h	??
qwt_plot_scaleitem.cpp	??
qwt_plot_scaleitem.h	??
qwt_plot_spectrogram.cpp	??
qwt_plot_spectrogram.h	??
qwt_plot_svgitem.cpp	??
qwt_plot_svgitem.h	??
qwt_plot_xml.cpp	??
qwt_plot_zoomer.cpp	??
qwt_plot_zoomer.h	??
qwt_polygon.h	??
qwt_raster_data.cpp	??
qwt_raster_data.h	??
qwt_rect.cpp	??
qwt_rect.h	??
qwt_round_scale_draw.cpp	??
qwt_round_scale_draw.h	??
qwt_scale_div.cpp	??
qwt_scale_div.h	??
qwt_scale_draw.cpp	??
qwt_scale_draw.h	??
qwt_scale_engine.cpp	??
qwt_scale_engine.h	??
qwt_scale_map.cpp	??
qwt_scale_map.h	??
qwt_scale_widget.cpp	??
qwt_scale_widget.h	??
qwt_slider.cpp	??

<code>qwt_slider.h</code>	??
<code>qwt_spline.cpp</code>	??
<code>qwt_spline.h</code>	??
<code>qwt_symbol.cpp</code>	??
<code>qwt_symbol.h</code>	??
<code>qwt_text.cpp</code>	??
<code>qwt_text.h</code>	??
<code>qwt_text_engine.cpp</code>	??
<code>qwt_text_engine.h</code>	??
<code>qwt_text_label.cpp</code>	??
<code>qwt_text_label.h</code>	??
<code>qwt_thermo.cpp</code>	??
<code>qwt_thermo.h</code>	??
<code>qwt_valuelist.h</code>	??
<code>qwt_wheel.cpp</code>	??
<code>qwt_wheel.h</code>	??

5 Qwt User's Guide Page Index

5.1 Qwt User's Guide Related Pages

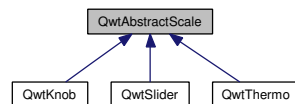
Here is a list of all related documentation pages:

Qwt License, Version 1.0	447
INSTALL	455
Curve Plots	458
Scatter Plot	458
Spectrogram, Contour Plot	458
Histogram	458
Dials, Compasses, Knobs, Wheels, Sliders, Thermos	458
Deprecated List	458
Todo List	458

6 Qwt User's Guide Class Documentation

6.1 QwtAbstractScale Class Reference

Inheritance diagram for QwtAbstractScale:



6.1.1 Detailed Description

An abstract base class for classes containing a scale.

[QwtAbstractScale](#) is used to provide classes with a [QwtScaleDraw](#), and a [QwtScaleDiv](#). The [QwtScaleDiv](#) might be set explicitly or calculated by a [QwtScaleEngine](#).

Definition at line 29 of file `qwt_abstract_scale.h`.

Public Member Functions

- [QwtAbstractScale](#) ()
- virtual [~QwtAbstractScale](#) ()
- void [setScale](#) (double vmin, double vmax, double step=0.0)
- void [setScale](#) (const [QwtDoubleInterval](#) &, double step=0.0)
- void [setScale](#) (const [QwtScaleDiv](#) &s)
- void [setAutoScale](#) ()
- bool [autoScale](#) () const
- void [setScaleMaxMajor](#) (int ticks)
- int [scaleMaxMinor](#) () const
- void [setScaleMaxMinor](#) (int ticks)
- int [scaleMaxMajor](#) () const
- void [setScaleEngine](#) ([QwtScaleEngine](#) *)
- const [QwtScaleEngine](#) * [scaleEngine](#) () const
- [QwtScaleEngine](#) * [scaleEngine](#) ()
- const [QwtScaleMap](#) & [scaleMap](#) () const

Protected Member Functions

- void [rescale](#) (double vmin, double vmax, double step=0.0)
- void [setAbstractScaleDraw](#) ([QwtAbstractScaleDraw](#) *)
- const [QwtAbstractScaleDraw](#) * [abstractScaleDraw](#) () const
- [QwtAbstractScaleDraw](#) * [abstractScaleDraw](#) ()
- virtual void [scaleChange](#) ()

6.1.2 Constructor & Destructor Documentation

6.1.2.1 QwtAbstractScale::QwtAbstractScale ()

Constructor

Creates a default [QwtScaleDraw](#) and a [QwtLinearScaleEngine](#). Autoscaling is enabled, and the *stepSize* is initialized by 0.0.

Definition at line 53 of file `qwt_abstract_scale.cpp`.

References `rescale()`.

6.1.2.2 QwtAbstractScale::~QwtAbstractScale () [virtual]

Destructor.

Definition at line 60 of file `qwt_abstract_scale.cpp`.

6.1.3 Member Function Documentation

6.1.3.1 void QwtAbstractScale::setScale (double *vmin*, double *vmax*, double *stepSize* = 0.0)

Specify a scale.

Disable autoscaling and define a scale by an interval and a step size

Parameters:

vmin lower limit of the scale interval

vmax upper limit of the scale interval

stepSize major step size

See also:

[setAutoScale\(\)](#)

Definition at line 75 of file `qwt_abstract_scale.cpp`.

References `rescale()`.

Referenced by `setScale()`.

6.1.3.2 void QwtAbstractScale::setScale (const [QwtDoubleInterval](#) & *interval*, double *stepSize* = 0.0)

Specify a scale.

Disable autoscaling and define a scale by an interval and a step size

Parameters:

interval Interval

stepSize major step size

See also:

[setAutoScale\(\)](#)

Definition at line 92 of file qwt_abstract_scale.cpp.

References QwtDoubleInterval::maxValue(), QwtDoubleInterval::minValue(), and setScale().

6.1.3.3 void QwtAbstractScale::setScale (const [QwtScaleDiv](#) & *scaleDiv*)

Specify a scale.

Disable autoscaling and define a scale by a scale division

Parameters:

scaleDiv Scale division

See also:

[setAutoScale\(\)](#)

Definition at line 107 of file qwt_abstract_scale.cpp.

References [scaleChange\(\)](#).

6.1.3.4 void QwtAbstractScale::setAutoScale ()

Advise the widget to control the scale range internally.

Autoscaling is on by default.

See also:

[setScale\(\)](#), [autoScale\(\)](#)

Definition at line 147 of file qwt_abstract_scale.cpp.

References [scaleChange\(\)](#).

6.1.3.5 bool QwtAbstractScale::autoScale () const

Returns:

`true` if autoscaling is enabled

Definition at line 159 of file qwt_abstract_scale.cpp.

Referenced by [QwtSlider::rangeChange\(\)](#), and [QwtThermo::setRange\(\)](#).

6.1.3.6 void QwtAbstractScale::setScaleMaxMajor (int *ticks*)

Set the maximum number of major tick intervals.

The scale's major ticks are calculated automatically such that the number of major intervals does not exceed ticks. The default value is 5.

Parameters:

ticks maximal number of major ticks.

See also:

[QwtAbstractScaleDraw](#)

Definition at line 173 of file qwt_abstract_scale.cpp.

6.1.3.7 int QwtAbstractScale::scaleMaxMinor () const

Returns:

Max. number of minor tick intervals The default value is 3.

Definition at line 204 of file qwt_abstract_scale.cpp.

6.1.3.8 void QwtAbstractScale::setScaleMaxMinor (int ticks)

Set the maximum number of minor tick intervals.

The scale's minor ticks are calculated automatically such that the number of minor intervals does not exceed ticks. The default value is 3.

Parameters:

ticks

See also:

[QwtAbstractScaleDraw](#)

Definition at line 191 of file qwt_abstract_scale.cpp.

6.1.3.9 int QwtAbstractScale::scaleMaxMajor () const

Returns:

Max. number of major tick intervals The default value is 5.

Definition at line 213 of file qwt_abstract_scale.cpp.

6.1.3.10 void QwtAbstractScale::setScaleEngine (QwtScaleEngine * scaleEngine)

Set a scale engine.

The scale engine is responsible for calculating the scale division, and in case of auto scaling how to align the scale.

scaleEngine has to be created with new and will be deleted in ~QwtAbstractScale or the next call of setScaleEngine.

Definition at line 269 of file qwt_abstract_scale.cpp.

References scaleEngine().

Referenced by QwtThermo::setRange().

6.1.3.11 const QwtScaleEngine * QwtAbstractScale::scaleEngine () const

Returns:

Scale engine

See also:

[setScaleEngine\(\)](#)

Definition at line 282 of file qwt_abstract_scale.cpp.

Referenced by QwtThermo::setRange(), and setScaleEngine().

6.1.3.12 [QwtScaleEngine](#) * QwtAbstractScale::scaleEngine ()

Returns:

Scale engine

See also:

[setScaleEngine\(\)](#)

Definition at line 291 of file qwt_abstract_scale.cpp.

6.1.3.13 `const QwtScaleMap & QwtAbstractScale::scaleMap () const`

Returns:

[abstractScaleDraw\(\)](#)->[scaleMap\(\)](#)

Definition at line 308 of file qwt_abstract_scale.cpp.

6.1.3.14 `void QwtAbstractScale::rescale (double vmin, double vmax, double stepSize = 0.0)` [protected]

Recalculate the scale division and update the scale draw.

Parameters:

vmin Lower limit of the scale interval
vmax Upper limit of the scale interval
stepSize Major step size

See also:

[scaleChange\(\)](#)

Definition at line 127 of file qwt_abstract_scale.cpp.

References [scaleChange\(\)](#).

Referenced by [QwtAbstractScale\(\)](#), [QwtSlider::rangeChange\(\)](#), [QwtThermo::setRange\(\)](#), and [setScale\(\)](#).

6.1.3.15 `void QwtAbstractScale::setAbstractScaleDraw (QwtAbstractScaleDraw * scaleDraw)` [protected]

Set a scale draw.

scaleDraw has to be created with `new` and will be deleted in `~QwtAbstractScale` or the next call of `setAbstractScaleDraw`.

Definition at line 224 of file qwt_abstract_scale.cpp.

References [QwtAbstractScaleDraw::setScaleDiv\(\)](#).

Referenced by [QwtThermo::setScaleDraw\(\)](#), [QwtSlider::setScaleDraw\(\)](#), and [QwtKnob::setScaleDraw\(\)](#).

6.1.3.16 `const QwtAbstractScaleDraw * QwtAbstractScale::abstractScaleDraw () const`
[protected]

Returns:

Scale draw

See also:

[setAbstractScaleDraw\(\)](#)

Definition at line 249 of file `qwt_abstract_scale.cpp`.

Referenced by `QwtThermo::scaleDraw()`, `QwtSlider::scaleDraw()`, and `QwtKnob::scaleDraw()`.

6.1.3.17 `QwtAbstractScaleDraw * QwtAbstractScale::abstractScaleDraw ()` [protected]

Returns:

Scale draw

See also:

[setAbstractScaleDraw\(\)](#)

Definition at line 240 of file `qwt_abstract_scale.cpp`.

6.1.3.18 `void QwtAbstractScale::scaleChange ()` [protected, virtual]

Notify changed scale.

Dummy empty implementation, intended to be overloaded by derived classes

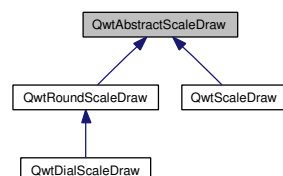
Reimplemented in [QwtSlider](#), and [QwtThermo](#).

Definition at line 301 of file `qwt_abstract_scale.cpp`.

Referenced by `rescale()`, `setAutoScale()`, and `setScale()`.

6.2 QwtAbstractScaleDraw Class Reference

Inheritance diagram for QwtAbstractScaleDraw:



6.2.1 Detailed Description

A abstract base class for drawing scales.

[QwtAbstractScaleDraw](#) can be used to draw linear or logarithmic scales.

After a scale division has been specified as a [QwtScaleDiv](#) object using [QwtAbstractScaleDraw::setScaleDiv\(const QwtScaleDiv &s\)](#), the scale can be drawn with the [QwtAbstractScaleDraw::draw\(\)](#) member.

Definition at line 37 of file `qwt_abstract_scale_draw.h`.

Public Types

- enum [ScaleComponent](#) {
Backbone = 1,
Ticks = 2,
Labels = 4 }

Public Member Functions

- [QwtAbstractScaleDraw](#) ()
- [QwtAbstractScaleDraw](#) (const [QwtAbstractScaleDraw](#) &)
- virtual [~QwtAbstractScaleDraw](#) ()
- [QwtAbstractScaleDraw](#) & operator= (const [QwtAbstractScaleDraw](#) &)
- void [setScaleDiv](#) (const [QwtScaleDiv](#) &s)
- const [QwtScaleDiv](#) & [scaleDiv](#) () const
- void [setTransformation](#) ([QwtScaleTransformation](#) *)
- const [QwtScaleMap](#) & [map](#) () const
- void [enableComponent](#) ([ScaleComponent](#), bool enable=true)
- bool [hasComponent](#) ([ScaleComponent](#)) const
- void [setTickLength](#) ([QwtScaleDiv::TickType](#), int length)
- int [tickLength](#) ([QwtScaleDiv::TickType](#)) const
- int [majTickLength](#) () const
- void [setSpacing](#) (int margin)
- int [spacing](#) () const
- virtual void [draw](#) (QPainter *, const QPalette &) const
- virtual [QwtText](#) [label](#) (double) const
- virtual int [extent](#) (const QPen &, const QFont &) const=0
- void [setMinimumExtent](#) (int)
- int [minimumExtent](#) () const
- [QwtScaleMap](#) & [scaleMap](#) ()

Protected Member Functions

- virtual void [drawTick](#) (QPainter *painter, double value, int len) const=0
- virtual void [drawBackbone](#) (QPainter *painter) const=0
- virtual void [drawLabel](#) (QPainter *painter, double value) const=0
- void [invalidateCache](#) ()
- const [QwtText](#) & [tickLabel](#) (const QFont &, double value) const

6.2.2 Member Enumeration Documentation

6.2.2.1 enum [QwtAbstractScaleDraw::ScaleComponent](#)

Components of a scale

- Backbone
- Ticks
- Labels

See also:

[QwtAbstractScaleDraw::enableComponent](#), [QwtAbstractScaleDraw::hasComponent](#)

Definition at line 52 of file `qwt_abstract_scale_draw.h`.

6.2.3 Constructor & Destructor Documentation

6.2.3.1 [QwtAbstractScaleDraw::QwtAbstractScaleDraw \(\)](#)

Constructor.

The range of the scale is initialized to [0, 100], The spacing (distance between ticks and labels) is set to 4, the tick lengths are set to 4,6 and 8 pixels

Definition at line 55 of file `qwt_abstract_scale_draw.cpp`.

6.2.3.2 [QwtAbstractScaleDraw::QwtAbstractScaleDraw \(const \[QwtAbstractScaleDraw\]\(#\) &\)](#)

Copy constructor.

Definition at line 61 of file `qwt_abstract_scale_draw.cpp`.

References `d_data`.

6.2.3.3 [QwtAbstractScaleDraw::~QwtAbstractScaleDraw \(\)](#) [virtual]

Destructor.

Definition at line 67 of file `qwt_abstract_scale_draw.cpp`.

6.2.4 Member Function Documentation

6.2.4.1 [QwtAbstractScaleDraw](#) & [QwtAbstractScaleDraw::operator= \(const \[QwtAbstractScaleDraw\]\(#\) &\)](#)

Assignment operator.

Definition at line 72 of file `qwt_abstract_scale_draw.cpp`.

References `d_data`.

6.2.4.2 void [QwtAbstractScaleDraw::setScaleDiv \(const \[QwtScaleDiv\]\(#\) & sd\)](#)

Change the scale division

Parameters:

sd New scale division

Definition at line 108 of file qwt_abstract_scale_draw.cpp.

References QwtScaleDiv::hBound(), and QwtScaleDiv::lBound().

Referenced by QwtAbstractScale::setAbstractScaleDraw(), QwtScaleWidget::setScaleDiv(), and QwtPlotScaleItem::updateScaleDiv().

6.2.4.3 const QwtScaleDiv & QwtAbstractScaleDraw::scaleDiv () const**Returns:**

scale division

Definition at line 138 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtRoundScaleDraw::extent(), QwtScaleDraw::getBorderDistHint(), QwtScaleDraw::maxLabelHeight(), QwtScaleDraw::maxLabelWidth(), QwtScaleDraw::minLabelDist(), QwtScaleDraw::minLength(), QwtScaleWidget::setScaleDiv(), and QwtPlot::sizeHint().

6.2.4.4 void QwtAbstractScaleDraw::setTransformation (QwtScaleTransformation * transformation)

Change the transformation of the scale

Parameters:

transformation New scale transformation

Definition at line 119 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtPlotScaleItem::draw(), and QwtScaleWidget::setScaleDiv().

6.2.4.5 const QwtScaleMap & QwtAbstractScaleDraw::map () const**Returns:**

Map how to translate between scale and pixel values

Definition at line 126 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtRoundScaleDraw::drawBackbone(), QwtScaleWidget::drawColorBar(), QwtRoundScaleDraw::drawLabel(), QwtScaleDraw::drawTick(), QwtRoundScaleDraw::drawTick(), QwtRoundScaleDraw::extent(), QwtScaleDraw::getBorderDistHint(), QwtScaleDraw::labelPosition(), and QwtScaleWidget::setScaleDiv().

6.2.4.6 void QwtAbstractScaleDraw::enableComponent (ScaleComponent component, bool enable = true)

En/Disable a component of the scale

Parameters:

component Scale component

enable On/Off

See also:

[QwtAbstractScaleDraw::hasComponent](#)

Definition at line 86 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtDial::setScaleOptions().

6.2.4.7 bool QwtAbstractScaleDraw::hasComponent ([ScaleComponent component](#)) const

Check if a component is enabled

See also:

[QwtAbstractScaleDraw::enableComponent](#)

Definition at line 99 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtRoundScaleDraw::drawLabel(), QwtScaleDraw::extent(), QwtRoundScaleDraw::extent(), QwtScaleDraw::getBorderDistHint(), QwtScaleDraw::labelPosition(), QwtScaleDraw::minLabelDist(), and QwtScaleDraw::minLength().

6.2.4.8 void QwtAbstractScaleDraw::setTickLength ([QwtScaleDiv::TickType tickType](#), int *length*)

Set the length of the ticks

Parameters:

tickType Tick type

length New length

Warning:

the length is limited to [0..1000]

Definition at line 306 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtDial::setScaleTicks().

6.2.4.9 int QwtAbstractScaleDraw::tickLength ([QwtScaleDiv::TickType tickType](#)) const

Return the length of the ticks

See also:

[QwtAbstractScaleDraw::setTickLength](#), [QwtAbstractScaleDraw::majTickLength](#)

Definition at line 331 of file qwt_abstract_scale_draw.cpp.

6.2.4.10 int QwtAbstractScaleDraw::majTickLength () const

The same as QwtAbstractScaleDraw::tickLength(QwtScaleDiv::MajorTick).

Definition at line 345 of file qwt_abstract_scale_draw.cpp.

Referenced by QwtRoundScaleDraw::drawLabel(), QwtScaleDraw::extent(), QwtRoundScaleDraw::extent(), and QwtScaleDraw::labelPosition().

6.2.4.11 void QwtAbstractScaleDraw::setSpacing (int *spacing*)

Set the spacing between tick and labels.

The spacing is the distance between ticks and labels. The default spacing is 4 pixels.

Parameters:

spacing Spacing

See also:

[QwtAbstractScaleDraw::spacing](#)

Definition at line 247 of file qwt_abstract_scale_draw.cpp.

6.2.4.12 int QwtAbstractScaleDraw::spacing () const

Get the spacing.

The spacing is the distance between ticks and labels. The default spacing is 4 pixels.

See also:

[QwtAbstractScaleDraw::setSpacing](#)

Definition at line 263 of file qwt_abstract_scale_draw.cpp.

Referenced by [QwtRoundScaleDraw::drawLabel\(\)](#), [QwtScaleDraw::extent\(\)](#), [QwtRoundScaleDraw::extent\(\)](#), and [QwtScaleDraw::labelPosition\(\)](#).

6.2.4.13 void QwtAbstractScaleDraw::draw (QPainter * *painter*, const QPalette & *palette*) const [virtual]

Draw the scale.

Parameters:

painter The painter

palette Palette, text color is used for the labels, foreground color for ticks and backbone

Definition at line 165 of file qwt_abstract_scale_draw.cpp.

Referenced by [QwtThermo::draw\(\)](#), [QwtSlider::draw\(\)](#), [QwtPlotScaleItem::draw\(\)](#), [QwtKnob::draw\(\)](#), and [QwtPlot::printScale\(\)](#).

6.2.4.14 [QwtText](#) QwtAbstractScaleDraw::label (double *value*) const [virtual]

Convert a value into its representing label.

The value is converted to a plain text using [QLocale::system\(\).toString\(value\)](#). This method is often overloaded by applications to have individual labels.

Parameters:

value Value

Returns:

Label string.

Reimplemented in [QwtDialScaleDraw](#).

Definition at line 361 of file `qwt_abstract_scale_draw.cpp`.

Referenced by `QwtRoundScaleDraw::drawLabel()`, `QwtRoundScaleDraw::extent()`, `QwtDialScaleDraw::label()`, and `tickLabel()`.

6.2.4.15 virtual int QwtAbstractScaleDraw::extent (const QPen &, const QFont &) const [pure virtual]

Calculate the extent

The extent is the distance from the baseline to the outermost pixel of the scale draw in opposite to its orientation. It is at least [minimumExtent\(\)](#) pixels.

See also:

[setMinimumExtent\(\)](#), [minimumExtent\(\)](#)

Implemented in [QwtRoundScaleDraw](#), and [QwtScaleDraw](#).

6.2.4.16 void QwtAbstractScaleDraw::setMinimumExtent (int *minExtent*)

Set a minimum for the extent.

The extent is calculated from the components of the scale draw. In situations, where the labels are changing and the layout depends on the extent (f.e scrolling a scale), setting an upper limit as minimum extent will avoid jumps of the layout.

Parameters:

minExtent Minimum extent

See also:

[extent\(\)](#), [minimumExtent\(\)](#)

Definition at line 281 of file `qwt_abstract_scale_draw.cpp`.

6.2.4.17 int QwtAbstractScaleDraw::minimumExtent () const

Get the minimum extent

See also:

[extent\(\)](#), [setMinimumExtent\(\)](#)

Definition at line 293 of file `qwt_abstract_scale_draw.cpp`.

Referenced by `QwtScaleDraw::extent()`, and `QwtRoundScaleDraw::extent()`.

6.2.4.18 [QwtScaleMap](#) & `QwtAbstractScaleDraw::scaleMap()`

Returns:

Map how to translate between scale and pixel values

Definition at line 132 of file `qwt_abstract_scale_draw.cpp`.

Referenced by `QwtScaleDraw::drawTick()`, `QwtRoundScaleDraw::QwtRoundScaleDraw()`, and `QwtRoundScaleDraw::setAngleRange()`.

6.2.4.19 `virtual void QwtAbstractScaleDraw::drawTick (QPainter * painter, double value, int len) const` [protected, pure virtual]

Draw a tick

Parameters:

painter Painter

value Value of the tick

len Length of the tick

See also:

[drawBackbone\(\)](#), [drawLabel\(\)](#)

Implemented in [QwtRoundScaleDraw](#), and [QwtScaleDraw](#).

6.2.4.20 `virtual void QwtAbstractScaleDraw::drawBackbone (QPainter * painter) const` [protected, pure virtual]

Draws the baseline of the scale

Parameters:

painter Painter

See also:

[drawTick\(\)](#), [drawLabel\(\)](#)

Implemented in [QwtRoundScaleDraw](#), and [QwtScaleDraw](#).

6.2.4.21 `virtual void QwtAbstractScaleDraw::drawLabel (QPainter * painter, double value) const` [protected, pure virtual]

Draws the label for a major scale tick

Parameters:

painter Painter

value Value

See also:

[drawTick](#), [drawBackbone](#)

Implemented in [QwtRoundScaleDraw](#), and [QwtScaleDraw](#).

6.2.4.22 void QwtAbstractScaleDraw::invalidateCache () [protected]

Invalidate the cache used by [QwtAbstractScaleDraw::tickLabel](#)

The cache is invalidated, when a new [QwtScaleDiv](#) is set. If the labels need to be changed, while the same [QwtScaleDiv](#) is set, [QwtAbstractScaleDraw::invalidateCache](#) needs to be called manually.

Definition at line 404 of file `qwt_abstract_scale_draw.cpp`.

6.2.4.23 const [QwtText](#) & QwtAbstractScaleDraw::tickLabel (const QFont & font, double value)
const [protected]

Convert a value into its representing label and cache it.

The conversion between value and label is called very often in the layout and painting code. Unfortunately the calculation of the label sizes might be slow (really slow for rich text in Qt4), so it's necessary to cache the labels.

Parameters:

font Font
value Value

Returns:

Tick label

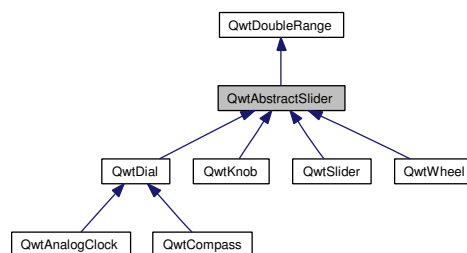
Definition at line 379 of file `qwt_abstract_scale_draw.cpp`.

References [label\(\)](#), and [QwtText::setRenderFlags\(\)](#).

Referenced by [QwtScaleDraw::boundingLabelRect\(\)](#), [QwtScaleDraw::drawLabel\(\)](#), [QwtRoundScaleDraw::drawLabel\(\)](#), [QwtRoundScaleDraw::extent\(\)](#), and [QwtScaleDraw::labelRect\(\)](#).

6.3 QwtAbstractSlider Class Reference

Inheritance diagram for QwtAbstractSlider:



Collaboration diagram for QwtAbstractSlider:



6.3.1 Detailed Description

An abstract base class for slider widgets.

[QwtAbstractSlider](#) is a base class for slider widgets. It handles mouse events and updates the slider's value accordingly. Derived classes only have to implement the [getValue\(\)](#) and [getScrollMode\(\)](#) members, and should react to a [valueChange\(\)](#), which normally requires repainting.

Definition at line 28 of file `qwt_abstract_slider.h`.

Public Types

- enum [ScrollMode](#) {
 ScrNone,
 ScrMouse,
 ScrTimer,
 ScrDirect,
 ScrPage }

Public Slots

- virtual void [setValue](#) (double val)
- virtual void [fitValue](#) (double val)
- virtual void [incValue](#) (int steps)
- virtual void [setReadOnly](#) (bool)

Signals

- void [valueChanged](#) (double value)
- void [sliderPressed](#) ()
- void [sliderReleased](#) ()
- void [sliderMoved](#) (double value)

Public Member Functions

- [QwtAbstractSlider](#) (Qt::Orientation, QWidget *parent=NULL)
- virtual [~QwtAbstractSlider](#) ()
- void [setUpdateTime](#) (int t)
- void [stopMoving](#) ()
- void [setTracking](#) (bool enable)
- virtual void [setMass](#) (double val)
- virtual double [mass](#) () const
- virtual void [setOrientation](#) (Qt::Orientation o)
- Qt::Orientation [orientation](#) () const
- bool [isReadOnly](#) () const
- bool [isValid](#) () const
- void [setValid](#) (bool valid)

Protected Member Functions

- virtual void [setPosition](#) (const QPoint &)
- virtual void [valueChange](#) ()
- virtual void [timerEvent](#) (QTimerEvent *e)
- virtual void [mousePressEvent](#) (QMouseEvent *e)
- virtual void [mouseReleaseEvent](#) (QMouseEvent *e)
- virtual void [mouseMoveEvent](#) (QMouseEvent *e)
- virtual void [keyPressEvent](#) (QKeyEvent *e)
- virtual void [wheelEvent](#) (QWheelEvent *e)
- virtual double [getValue](#) (const QPoint &p)=0
- virtual void [getScrollMode](#) (const QPoint &p, int &scrollMode, int &direction)=0
- void [setMouseOffset](#) (double)
- double [mouseOffset](#) () const
- int [scrollMode](#) () const

6.3.2 Member Enumeration Documentation

6.3.2.1 enum [QwtAbstractSlider::ScrollMode](#)

Scroll mode

See also:

[getScrollMode\(\)](#)

Definition at line 50 of file `qwt_abstract_slider.h`.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 [QwtAbstractSlider::QwtAbstractSlider](#) (Qt::Orientation *orientation*, QWidget * *parent* = NULL) [explicit]

Constructor.

Parameters:

orientation Orientation

parent Parent widget

Definition at line 54 of file `qwt_abstract_slider.cpp`.

6.3.3.2 [QwtAbstractSlider::~~QwtAbstractSlider](#) () [virtual]

Destructor.

Definition at line 68 of file `qwt_abstract_slider.cpp`.

6.3.4 Member Function Documentation

6.3.4.1 void [QwtAbstractSlider::setUpdateTime](#) (int *t*)

Specify the update interval for automatic scrolling.

Parameters:

t update interval in milliseconds

See also:

[getScrollMode\(\)](#)

Definition at line 138 of file qwt_abstract_slider.cpp.

6.3.4.2 void QwtAbstractSlider::stopMoving ()

Stop updating if automatic scrolling is active.

Definition at line 124 of file qwt_abstract_slider.cpp.

Referenced by [fitValue\(\)](#), [incValue\(\)](#), [mousePressEvent\(\)](#), [mouseReleaseEvent\(\)](#), [setValue\(\)](#), and [timerEvent\(\)](#).

6.3.4.3 void QwtAbstractSlider::setTracking (bool *enable*)

Enables or disables tracking.

If tracking is enabled, the slider emits a [valueChanged\(\)](#) signal whenever its value changes (the default behaviour). If tracking is disabled, the value changed() signal will only be emitted if:

- the user releases the mouse button and the value has changed or
- at the end of automatic scrolling.

Tracking is enabled by default.

Parameters:

enable `true` (enable) or `false` (disable) tracking.

Definition at line 296 of file qwt_abstract_slider.cpp.

6.3.4.4 void QwtAbstractSlider::setMass (double *val*) [virtual]

Set the slider's mass for flywheel effect.

If the slider's mass is greater than 0, it will continue to move after the mouse button has been released. Its speed decreases with time at a rate depending on the slider's mass. A large mass means that it will continue to move for a long time.

Derived widgets may overload this function to make it public.

Parameters:

val New mass in kg

See also:

[mass\(\)](#)

Reimplemented in [QwtWheel](#).

Definition at line 507 of file qwt_abstract_slider.cpp.

Referenced by [QwtWheel::setMass\(\)](#).

6.3.4.5 double QwtAbstractSlider::mass () const [virtual]

Returns:

mass

See also:

[setMass\(\)](#)

Reimplemented in [QwtWheel](#).

Definition at line 521 of file qwt_abstract_slider.cpp.

Referenced by [QwtWheel::mass\(\)](#).

6.3.4.6 void QwtAbstractSlider::setOrientation (Qt::Orientation o) [virtual]

Set the orientation.

Parameters:

o Orientation. Allowed values are Qt::Horizontal and Qt::Vertical.

Reimplemented in [QwtSlider](#), and [QwtWheel](#).

Definition at line 108 of file qwt_abstract_slider.cpp.

Referenced by [QwtWheel::setOrientation\(\)](#), and [QwtSlider::setOrientation\(\)](#).

6.3.4.7 Qt::Orientation QwtAbstractSlider::orientation () const

Returns:

Orientation

See also:

[setOrientation\(\)](#)

Definition at line 117 of file qwt_abstract_slider.cpp.

Referenced by [QwtSlider::drawSlider\(\)](#), [QwtSlider::drawThumb\(\)](#), [QwtWheel::drawWheel\(\)](#), [QwtWheel::drawWheelBackground\(\)](#), [QwtSlider::getScrollMode\(\)](#), [QwtWheel::getValue\(\)](#), [QwtSlider::getValue\(\)](#), [keyPressEvent\(\)](#), [QwtSlider::layoutSlider\(\)](#), [QwtWheel::minimumSizeHint\(\)](#), [QwtSlider::minimumSizeHint\(\)](#), [QwtWheel::setOrientation\(\)](#), and [QwtSlider::setOrientation\(\)](#).

6.3.4.8 bool QwtAbstractSlider::isReadOnly () const

In read only mode the slider can't be controlled by mouse or keyboard.

Returns:

true if read only

See also:

[setReadOnly\(\)](#)

Definition at line 98 of file qwt_abstract_slider.cpp.

Referenced by QwtDial::drawFocusIndicator(), QwtDial::keyPressEvent(), QwtCompass::keyPressEvent(), keyPressEvent(), mouseMoveEvent(), mousePressEvent(), mouseReleaseEvent(), and wheelEvent().

6.3.4.9 bool QwtAbstractSlider::isValid () const [inline]

See also:

QwtDblRange::isValid

Reimplemented from [QwtDoubleRange](#).

Definition at line 87 of file qwt_abstract_slider.h.

References QwtDoubleRange::isValid().

Referenced by QwtDial::drawContents(), QwtKnob::drawKnob(), QwtAnalogClock::drawNeedle(), QwtCompass::drawScaleContents(), QwtSlider::drawSlider(), QwtDial::keyPressEvent(), keyPressEvent(), mouseMoveEvent(), mousePressEvent(), mouseReleaseEvent(), and wheelEvent().

6.3.4.10 void QwtAbstractSlider::setValid (bool *valid*) [inline]

See also:

QwtDblRange::isValid

Reimplemented from [QwtDoubleRange](#).

Definition at line 92 of file qwt_abstract_slider.h.

References QwtDoubleRange::setValid().

Referenced by QwtAnalogClock::setTime().

6.3.4.11 void QwtAbstractSlider::setValue (double *val*) [virtual, slot]

Move the slider to a specified value.

This function can be used to move the slider to a value which is not an integer multiple of the step size.

Parameters:

val new value

See also:

[fitValue\(\)](#)

Reimplemented from [QwtDoubleRange](#).

Definition at line 535 of file qwt_abstract_slider.cpp.

References QwtDoubleRange::setValue(), and stopMoving().

Referenced by QwtDial::keyPressEvent(), QwtCompass::keyPressEvent(), and QwtAnalogClock::setTime().

6.3.4.12 void QwtAbstractSlider::fitValue (double *value*) [virtual, slot]

Set the slider's value to the nearest integer multiple of the step size.

Parameters:

value Value

See also:

[setValue\(\)](#), [incValue\(\)](#)

Reimplemented from [QwtDoubleRange](#).

Definition at line 550 of file qwt_abstract_slider.cpp.

References [QwtDoubleRange::fitValue\(\)](#), and [stopMoving\(\)](#).

6.3.4.13 void QwtAbstractSlider::incValue (int *steps*) [virtual, slot]

Increment the value by a specified number of steps.

Parameters:

steps number of steps

See also:

[setValue\(\)](#)

Reimplemented from [QwtDoubleRange](#).

Definition at line 562 of file qwt_abstract_slider.cpp.

References [QwtDoubleRange::incValue\(\)](#), and [stopMoving\(\)](#).

6.3.4.14 void QwtAbstractSlider::setReadOnly (bool *readOnly*) [virtual, slot]

En/Disable read only mode

In read only mode the slider can't be controlled by mouse or keyboard.

Parameters:

readOnly Enables in case of true

See also:

[isReadOnly\(\)](#)

Definition at line 85 of file qwt_abstract_slider.cpp.

6.3.4.15 void QwtAbstractSlider::valueChanged (double *value*) [signal]

Notify a change of value.

In the default setting (tracking enabled), this signal will be emitted every time the value changes (see [setTracking\(\)](#)).

Parameters:

value new value

Referenced by `valueChange()`.

6.3.4.16 void QwtAbstractSlider::sliderPressed () [signal]

This signal is emitted when the user presses the movable part of the slider (start ScrMouse Mode).

Referenced by `mousePressEvent()`.

6.3.4.17 void QwtAbstractSlider::sliderReleased () [signal]

This signal is emitted when the user releases the movable part of the slider.

Referenced by `mouseReleaseEvent()`.

6.3.4.18 void QwtAbstractSlider::sliderMoved (double *value*) [signal]

This signal is emitted when the user moves the slider with the mouse.

Parameters:

value new value

Referenced by `QwtDial::keyPressEvent()`, `keyPressEvent()`, `mouseMoveEvent()`, and `wheelEvent()`.

6.3.4.19 void QwtAbstractSlider::setPosition (const QPoint & *p*) [protected, virtual]

Move the slider to a specified point, adjust the value and emit signals if necessary.

Definition at line 276 of file `qwt_abstract_slider.cpp`.

References `QwtDoubleRange::fitValue()`, and `getValue()`.

Referenced by `mouseMoveEvent()`, and `mouseReleaseEvent()`.

6.3.4.20 void QwtAbstractSlider::valueChange () [protected, virtual]

Notify change of value

This function can be reimplemented by derived classes in order to keep track of changes, i.e. repaint the widget. The default implementation emits a `valueChanged()` signal if tracking is enabled.

Reimplemented from [QwtDoubleRange](#).

Reimplemented in [QwtDial](#), [QwtSlider](#), and [QwtWheel](#).

Definition at line 484 of file `qwt_abstract_slider.cpp`.

References `QwtDoubleRange::value()`, and `valueChanged()`.

Referenced by `QwtWheel::valueChange()`, `QwtSlider::valueChange()`, and `QwtDial::valueChange()`.

6.3.4.21 void QwtAbstractSlider::timerEvent (QTimerEvent * *e*) [protected, virtual]

Qt timer event

Parameters:

e Timer event

Definition at line 417 of file qwt_abstract_slider.cpp.

References QwtDoubleRange::exactValue(), QwtDoubleRange::fitValue(), QwtDoubleRange::incPages(), QwtDoubleRange::step(), stopMoving(), and QwtDoubleRange::value().

6.3.4.22 void QwtAbstractSlider::mousePressEvent (QMouseEvent * *e*) [protected, virtual]

Mouse press event handler.

Definition at line 147 of file qwt_abstract_slider.cpp.

References getScrollMode(), getValue(), isReadOnly(), isValid(), sliderPressed(), stopMoving(), and QwtDoubleRange::value().

6.3.4.23 void QwtAbstractSlider::mouseReleaseEvent (QMouseEvent * *e*) [protected, virtual]

Mouse Release Event handler.

Definition at line 196 of file qwt_abstract_slider.cpp.

References QwtDoubleRange::fitValue(), QwtDoubleRange::incPages(), isReadOnly(), isValid(), setPosition(), sliderReleased(), QwtDoubleRange::step(), stopMoving(), and QwtDoubleRange::value().

6.3.4.24 void QwtAbstractSlider::mouseMoveEvent (QMouseEvent * *e*) [protected, virtual]

Mouse Move Event handler

Parameters:

e Mouse event

Definition at line 305 of file qwt_abstract_slider.cpp.

References QwtDoubleRange::exactPrevValue(), QwtDoubleRange::exactValue(), isReadOnly(), isValid(), QwtDoubleRange::prevValue(), setPosition(), sliderMoved(), and QwtDoubleRange::value().

6.3.4.25 void QwtAbstractSlider::keyPressEvent (QKeyEvent * *e*) [protected, virtual]

Handles key events

- Key_Down, Key_Left
Decrement by 1
- Key_Up, Key_Right
Increment by 1

Parameters:

e Key event

See also:

[isReadOnly\(\)](#)

Reimplemented in [QwtCompass](#), and [QwtDial](#).

Definition at line 371 of file `qwt_abstract_slider.cpp`.

References `QwtDoubleRange::incValue()`, `isReadOnly()`, `isValid()`, `orientation()`, `QwtDoubleRange::prevValue()`, `sliderMoved()`, and `QwtDoubleRange::value()`.

6.3.4.26 void QwtAbstractSlider::wheelEvent (QWheelEvent * *e*) [protected, virtual]

Wheel Event handler

Parameters:

e Wheel event

Definition at line 336 of file `qwt_abstract_slider.cpp`.

References `getScrollMode()`, `QwtDoubleRange::incPages()`, `isReadOnly()`, `isValid()`, `QwtDoubleRange::prevValue()`, `sliderMoved()`, and `QwtDoubleRange::value()`.

6.3.4.27 virtual double QwtAbstractSlider::getValue (const QPoint & *p*) [protected, pure virtual]

Determine the value corresponding to a specified point.

This is an abstract virtual function which is called when the user presses or releases a mouse button or moves the mouse. It has to be implemented by the derived class.

Parameters:

p point

Implemented in [QwtDial](#), [QwtSlider](#), and [QwtWheel](#).

Referenced by `mousePressEvent()`, and `setPosition()`.

6.3.4.28 virtual void QwtAbstractSlider::getScrollMode (const QPoint & *p*, int & *scrollMode*, int & *direction*) [protected, pure virtual]

Determine what to do when the user presses a mouse button.

This function is abstract and has to be implemented by derived classes. It is called on a `mousePressEvent`. The derived class can determine what should happen next in dependence of the position where the mouse was pressed by returning scrolling mode and direction. [QwtAbstractSlider](#) knows the following modes:

QwtAbstractSlider::ScrNone Scrolling switched off. Don't change the value.

QwtAbstractSlider::ScrMouse Change the value while the user keeps the button pressed and moves the mouse.

QwtAbstractSlider::ScrTimer Automatic scrolling. Increment the value in the specified direction as long as the user keeps the button pressed.

QwtAbstractSlider::ScrPage Automatic scrolling. Same as `ScrTimer`, but increment by page size.

Parameters:

p point where the mouse was pressed

Return values:

scrollMode The scrolling mode

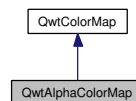
direction direction: 1, 0, or -1.

Implemented in [QwtDial](#), [QwtSlider](#), and [QwtWheel](#).

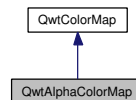
Referenced by `mousePressEvent()`, and `wheelEvent()`.

6.4 QwtAlphaColorMap Class Reference

Inheritance diagram for QwtAlphaColorMap:



Collaboration diagram for QwtAlphaColorMap:



6.4.1 Detailed Description

[QwtAlphaColorMap](#) varies the alpha value of a color.

Definition at line 160 of file `qwt_color_map.h`.

Public Member Functions

- [QwtAlphaColorMap](#) (const QColor &=QColor(Qt::gray))
- [QwtAlphaColorMap](#) (const [QwtAlphaColorMap](#) &)
- virtual [~QwtAlphaColorMap](#) ()
- [QwtAlphaColorMap](#) & [operator=](#) (const [QwtAlphaColorMap](#) &)
- virtual [QwtColorMap](#) * [copy](#) () const
- void [setColor](#) (const QColor &)
- QColor [color](#) () const
- virtual QRgb [rgb](#) (const [QwtDoubleInterval](#) &, double value) const

6.4.2 Constructor & Destructor Documentation

6.4.2.1 [QwtAlphaColorMap::QwtAlphaColorMap](#) (const QColor & *color* = QColor(Qt::gray))

Constructor

Parameters:

color Color of the map

Definition at line 409 of file qwt_color_map.cpp.

Referenced by copy().

6.4.2.2 QwtAlphaColorMap::QwtAlphaColorMap (const [QwtAlphaColorMap](#) & *other*)

Copy constructor

Parameters:

other Other color map

Definition at line 421 of file qwt_color_map.cpp.

6.4.2.3 QwtAlphaColorMap::~QwtAlphaColorMap () [virtual]

Destructor.

Definition at line 429 of file qwt_color_map.cpp.

6.4.3 Member Function Documentation**6.4.3.1 [QwtAlphaColorMap](#) & QwtAlphaColorMap::operator= (const [QwtAlphaColorMap](#) & *other*)**

Assignment operator

Parameters:

other Other color map

Returns:

*this

Definition at line 439 of file qwt_color_map.cpp.

References d_data.

6.4.3.2 [QwtColorMap](#) * QwtAlphaColorMap::copy () const [virtual]

Clone the color map.

Implements [QwtColorMap](#).

Definition at line 448 of file qwt_color_map.cpp.

References QwtAlphaColorMap().

6.4.3.3 void QwtAlphaColorMap::setColor (const QColor & *color*)

Set the color

Parameters:

color Color

See also:

[color\(\)](#)

Definition at line 462 of file qwt_color_map.cpp.

6.4.3.4 QColor QwtAlphaColorMap::color () const

Returns:

the color

See also:

[setColor\(\)](#)

Definition at line 472 of file qwt_color_map.cpp.

6.4.3.5 QRgb QwtAlphaColorMap::rgb (const [QwtDoubleInterval](#) & *interval*, double *value*) const [virtual]

Map a value of a given interval into a alpha value.

$\alpha := (\text{value} - \text{interval.minValue}()) / \text{interval.width}();$

Parameters:

interval Range for all values

value Value to map into a rgb value

Returns:

rgb value, with an alpha value

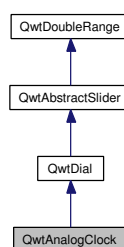
Implements [QwtColorMap](#).

Definition at line 486 of file qwt_color_map.cpp.

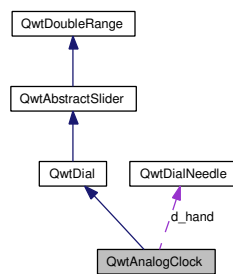
References [QwtDoubleInterval::minValue\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.5 QwtAnalogClock Class Reference

Inheritance diagram for QwtAnalogClock:



Collaboration diagram for QwtAnalogClock:



6.5.1 Detailed Description

An analog clock.

Example

```

#include <qwt_analog_clock.h>

QwtAnalogClock *clock = new QwtAnalogClock(...);
clock->scaleDraw()->setPenWidth(3);
clock->setLineWidth(6);
clock->setFrameShadow(QwtDial::Sunken);
clock->setTime();

// update the clock every second
QTimer *timer = new QTimer(clock);
timer->connect(timer, SIGNAL(timeout()), clock, SLOT(setCurrentTime()));
timer->start(1000);

```

Qwt is missing a set of good looking hands. Contributions are very welcome.

Note:

The examples/dials example shows how to use [QwtAnalogClock](#).

Definition at line 45 of file qwt_analog_clock.h.

Public Types

- enum [Hand](#) {
SecondHand,
MinuteHand,
HourHand,
NHands }

Public Slots

- void [setCurrentTime](#) ()
- void [setTime](#) (const QTime &=QTime::currentTime())

Public Member Functions

- [QwtAnalogClock](#) (QWidget *parent=NULL)
- virtual [~QwtAnalogClock](#) ()
- virtual void [setHand](#) ([Hand](#), [QwtDialNeedle](#) *)
- const [QwtDialNeedle](#) * [hand](#) ([Hand](#)) const
- [QwtDialNeedle](#) * [hand](#) ([Hand](#))

Protected Member Functions

- virtual [QwtText](#) [scaleLabel](#) (double) const
- virtual void [drawNeedle](#) (QPainter *, const QPoint &, int radius, double direction, QPalette::ColorGroup) const
- virtual void [drawHand](#) (QPainter *, [Hand](#), const QPoint &, int radius, double direction, QPalette::ColorGroup) const

6.5.2 Member Enumeration Documentation

6.5.2.1 enum [QwtAnalogClock::Hand](#)

Hand type

See also:

[setHand\(\)](#), [hand\(\)](#)

Definition at line 55 of file qwt_analog_clock.h.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 [QwtAnalogClock::QwtAnalogClock](#) (QWidget **parent* = NULL) [explicit]

Constructor

Parameters:

parent Parent widget

Definition at line 16 of file qwt_analog_clock.cpp.

6.5.3.2 [QwtAnalogClock::~QwtAnalogClock](#) () [virtual]

Destructor.

Definition at line 82 of file qwt_analog_clock.cpp.

6.5.4 Member Function Documentation

6.5.4.1 void [QwtAnalogClock::setHand](#) ([Hand](#) *hand*, [QwtDialNeedle](#) **needle*) [virtual]

Set a clockhand

Parameters:

hand Specifies the type of hand

needle Hand

See also:

[QwtAnalogClock::hand\(\)](#)

Definition at line 104 of file qwt_analog_clock.cpp.

References [QwtDial::needle\(\)](#).

6.5.4.2 `const QwtDialNeedle * QwtAnalogClock::hand (Hand hd) const`

Returns:

Clock hand

Parameters:

hd Specifies the type of hand

See also:

[QwtAnalogClock::setHand](#)

Definition at line 131 of file qwt_analog_clock.cpp.

Referenced by [drawHand\(\)](#).

6.5.4.3 `QwtDialNeedle * QwtAnalogClock::hand (Hand hd)`

Returns:

Clock hand

Parameters:

hd Specifies the type of hand

See also:

[QwtAnalogClock::setHand](#)

Definition at line 118 of file qwt_analog_clock.cpp.

6.5.4.4 `void QwtAnalogClock::setCurrentTime () [slot]`

Set the current time.

This is the same as [QwtAnalogClock::setTime\(\)](#), but Qt < 3.0 can't handle default parameters for slots.

Definition at line 142 of file qwt_analog_clock.cpp.

References [setTime\(\)](#).

6.5.4.5 `void QwtAnalogClock::setTime (const QTime & time = QTime::currentTime())`
[slot]

Set a time

Parameters:

time Time to display

Definition at line 151 of file qwt_analog_clock.cpp.

References `QwtAbstractSlider::setValid()`, and `QwtAbstractSlider::setValue()`.

Referenced by `setCurrentTime()`.

6.5.4.6 `QwtText QwtAnalogClock::scaleLabel (double value) const` [protected, virtual]

Find the scale label for a given value

Parameters:

value Value

Returns:

Label

Reimplemented from [QwtDial](#).

Definition at line 168 of file qwt_analog_clock.cpp.

6.5.4.7 `void QwtAnalogClock::drawNeedle (QPainter * painter, const QPoint & center, int radius, double direction, QPalette::ColorGroup cg) const` [protected, virtual]

Draw the needle.

A clock has no single needle but three hands instead. `drawNeedle` translates `value()` into directions for the hands and calls `drawHand()`.

Parameters:

painter Painter

center Center of the clock

radius Maximum length for the hands

direction Dummy, not used.

cg ColorGroup

See also:

[QwtAnalogClock::drawHand\(\)](#)

Reimplemented from [QwtDial](#).

Definition at line 191 of file qwt_analog_clock.cpp.

References `drawHand()`, `QwtAbstractSlider::isValid()`, `QwtDial::origin()`, and `QwtDoubleRange::value()`.

6.5.4.8 void QwtAnalogClock::drawHand (QPainter * *painter*, [Hand](#) *hd*, const QPoint & *center*, int *radius*, double *direction*, QPalette::ColorGroup *cg*) const [protected, virtual]

Draw a clock hand

Parameters:

painter Painter

hd Specify the type of hand

center Center of the clock

radius Maximum length for the hands

direction Direction of the hand in degrees, counter clockwise

cg ColorGroup

Definition at line 220 of file qwt_analog_clock.cpp.

References QwtDialNeedle::draw(), hand(), and QwtDial::needle().

Referenced by drawNeedle().

6.6 QwtArrayData Class Reference

Inheritance diagram for QwtArrayData:



Collaboration diagram for QwtArrayData:



6.6.1 Detailed Description

Data class containing two QwtArray<double> objects.

Definition at line 119 of file qwt_data.h.

Public Member Functions

- [QwtArrayData](#) (const QwtArray< double > &*x*, const QwtArray< double > &*y*)
- [QwtArrayData](#) (const double **x*, const double **y*, size_t *size*)
- [QwtArrayData](#) & [operator=](#) (const [QwtArrayData](#) &)
- virtual [QwtData](#) * [copy](#) () const
- virtual size_t [size](#) () const
- virtual double [x](#) (size_t *i*) const

- virtual double [y](#) (size_t i) const
- const QwtArray< double > & [xData](#) () const
- const QwtArray< double > & [yData](#) () const
- virtual [QwtDoubleRect boundingRect](#) () const

6.6.2 Constructor & Destructor Documentation

6.6.2.1 QwtArrayData::QwtArrayData (const QwtArray< double > & x, const QwtArray< double > & y)

Constructor

Parameters:

- x* Array of x values
- y* Array of y values

See also:

[QwtPlotCurve::setData](#)

Definition at line 141 of file qwt_data.cpp.

Referenced by [copy\(\)](#).

6.6.2.2 QwtArrayData::QwtArrayData (const double * x, const double * y, size_t size)

Constructor

Parameters:

- x* Array of x values
- y* Array of y values
- size* Size of the x and y arrays

See also:

[QwtPlotCurve::setData](#)

Definition at line 156 of file qwt_data.cpp.

6.6.3 Member Function Documentation

6.6.3.1 [QwtArrayData](#) & QwtArrayData::operator= (const [QwtArrayData](#) &)

Assignment.

Definition at line 174 of file qwt_data.cpp.

References [d_x](#), and [d_y](#).

6.6.3.2 [QwtData](#) * [QwtArrayData::copy \(\) const](#) [virtual]**Returns:**

Pointer to a copy (virtual copy constructor)

Implements [QwtData](#).

Definition at line 227 of file qwt_data.cpp.

References [QwtArrayData\(\)](#).

6.6.3.3 `size_t` [QwtArrayData::size \(\) const](#) [virtual]**Returns:**

Size of the data set

Implements [QwtData](#).

Definition at line 185 of file qwt_data.cpp.

Referenced by [boundingRect\(\)](#).

6.6.3.4 `double` [QwtArrayData::x \(size_t i\) const](#) [virtual]

Return the x value of data point i

Parameters:

i Index

Returns:

x X value of data point i

Implements [QwtData](#).

Definition at line 196 of file qwt_data.cpp.

6.6.3.5 `double` [QwtArrayData::y \(size_t i\) const](#) [virtual]

Return the y value of data point i

Parameters:

i Index

Returns:

y Y value of data point i

Implements [QwtData](#).

Definition at line 207 of file qwt_data.cpp.

6.6.3.6 `const QwtArray< double > & QwtArrayData::xData () const`**Returns:**

Array of the x-values

Definition at line 213 of file qwt_data.cpp.

6.6.3.7 `const QwtArray< double > & QwtArrayData::yData () const`**Returns:**

Array of the y-values

Definition at line 219 of file qwt_data.cpp.

6.6.3.8 `QwtDoubleRect QwtArrayData::boundingRect () const` [virtual]

Returns the bounding rectangle of the data. If there is no bounding rect, like for empty data the rectangle is invalid: `QwtDoubleRect::isValid() == false`

Reimplemented from [QwtData](#).

Definition at line 237 of file qwt_data.cpp.

References `size()`.

6.7 QwtArrowButton Class Reference

6.7.1 Detailed Description

Arrow Button.

A push button with one or more filled triangles on its front. An Arrow button can have 1 to 3 arrows in a row, pointing up, down, left or right.

Definition at line 23 of file qwt_arrow_button.h.

Public Member Functions

- [QwtArrowButton](#) (int num, Qt::ArrowType, QWidget *parent=NULL)
- virtual [~QwtArrowButton](#) ()
- Qt::ArrowType [arrowType](#) () const
- int [num](#) () const
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const

Protected Member Functions

- virtual void [paintEvent](#) (QPaintEvent *event)
- virtual void [drawButtonLabel](#) (QPainter *p)
- virtual void [drawArrow](#) (QPainter *, const QRect &, Qt::ArrowType) const
- virtual QRect [labelRect](#) () const
- virtual QSize [arrowSize](#) (Qt::ArrowType, const QSize &boundingSize) const
- virtual void [keyPressEvent](#) (QKeyEvent *)

6.7.2 Constructor & Destructor Documentation

6.7.2.1 QwtArrowButton::QwtArrowButton (int *num*, Qt::ArrowType *arrowType*, QWidget * *parent* = NULL) [explicit]

Parameters:

num Number of arrows

arrowType see Qt::ArrowType in the Qt docs.

parent Parent widget

Definition at line 58 of file qwt_arrow_button.cpp.

6.7.2.2 QwtArrowButton::~QwtArrowButton () [virtual]

Destructor.

Definition at line 83 of file qwt_arrow_button.cpp.

6.7.3 Member Function Documentation

6.7.3.1 Qt::ArrowType QwtArrowButton::arrowType () const

The direction of the arrows.

Definition at line 92 of file qwt_arrow_button.cpp.

6.7.3.2 int QwtArrowButton::num () const

The number of arrows.

Definition at line 100 of file qwt_arrow_button.cpp.

6.7.3.3 QSize QwtArrowButton::sizeHint () const [virtual]

Returns:

a size hint

Definition at line 286 of file qwt_arrow_button.cpp.

References `minimumSizeHint()`.

6.7.3.4 QSize QwtArrowButton::minimumSizeHint () const [virtual]

Return a minimum size hint.

Definition at line 294 of file qwt_arrow_button.cpp.

References `arrowSize()`.

Referenced by `sizeHint()`.

6.7.3.5 void QwtArrowButton::paintEvent (QPaintEvent * *event*) [protected, virtual]

Paint event handler

Parameters:

event Paint event

Definition at line 143 of file qwt_arrow_button.cpp.

References drawButtonLabel().

6.7.3.6 void QwtArrowButton::drawButtonLabel (QPainter * *painter*) [protected, virtual]

Draw the button label.

Parameters:

painter Painter

See also:

The Qt Manual on QPushButton

Definition at line 157 of file qwt_arrow_button.cpp.

References arrowSize(), drawArrow(), and labelRect().

Referenced by paintEvent().

6.7.3.7 void QwtArrowButton::drawArrow (QPainter * *painter*, const QRect & *r*, Qt::ArrowType *arrowType*) const [protected, virtual]

Draw an arrow int a bounding rect

Parameters:

painter Painter

r Rectangle where to paint the arrow

arrowType Arrow type

Definition at line 240 of file qwt_arrow_button.cpp.

Referenced by drawButtonLabel().

6.7.3.8 QRect QwtArrowButton::labelRect () const [protected, virtual]**Returns:**

the bounding rect for the label

Definition at line 108 of file qwt_arrow_button.cpp.

Referenced by drawButtonLabel().

6.7.3.9 QSize QwtArrowButton::arrowSize (Qt::ArrowType *arrowType*, const QSize & *boundingSize*) const [protected, virtual]

Calculate the size for a arrow that fits into a rect of a given size

Parameters:

arrowType Arrow type
boundingSize Bounding size

Returns:

Size of the arrow

Definition at line 331 of file qwt_arrow_button.cpp.

Referenced by drawButtonLabel(), and minimumSizeHint().

6.7.3.10 void QwtArrowButton::keyPressEvent (QKeyEvent *) [protected, virtual]

autoRepeat for the space keys

Definition at line 361 of file qwt_arrow_button.cpp.

6.8 QwtClipper Class Reference

6.8.1 Detailed Description

Some clipping algos.

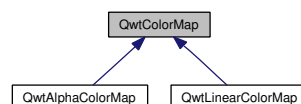
Definition at line 25 of file qwt_clipper.h.

Static Public Member Functions

- static QwtPolygon [clipPolygon](#) (const QRect &, const QwtPolygon &)
- static QwtPolygonF [clipPolygonF](#) (const QwtDoubleRect &, const QwtPolygonF &)
- static QwtArray< [QwtDoubleInterval](#) > [clipCircle](#) (const QwtDoubleRect &, const [QwtDoublePoint](#) &, double radius)

6.9 QwtColorMap Class Reference

Inheritance diagram for QwtColorMap:



6.9.1 Detailed Description

[QwtColorMap](#) is used to map values into colors.

For displaying 3D data on a 2D plane the 3rd dimension is often displayed using colors, like f.e in a spectrogram.

Each color map is optimized to return colors for only one of the following image formats:

- QImage::Format_Indexed8
- QImage::Format_ARGB32

See also:

[QwtPlotSpectrogram](#), [QwtScaleWidget](#)

Definition at line 44 of file qwt_color_map.h.

Public Types

- enum [Format](#) {
 RGB,
 Indexed }

Public Member Functions

- [QwtColorMap](#) ([Format](#)=QwtColorMap::RGB)
- virtual [~QwtColorMap](#) ()
- [Format](#) [format](#) () const
- virtual [QwtColorMap](#) * [copy](#) () const=0
- virtual [QRgb](#) [rgb](#) (const [QwtDoubleInterval](#) &interval, double value) const=0
- virtual unsigned char [colorIndex](#) (const [QwtDoubleInterval](#) &interval, double value) const=0
- [QColor](#) [color](#) (const [QwtDoubleInterval](#) &, double value) const
- virtual [QVector](#)< [QRgb](#) > [colorTable](#) (const [QwtDoubleInterval](#) &) const

6.9.2 Member Enumeration Documentation

6.9.2.1 enum [QwtColorMap::Format](#)

- RGB
The map is intended to map into [QRgb](#) values.
- Indexed
The map is intended to map into 8 bit values, that are indices into the color table.

See also:

[rgb\(\)](#), [colorIndex\(\)](#), [colorTable\(\)](#)

Definition at line 57 of file qwt_color_map.h.

6.9.3 Constructor & Destructor Documentation

6.9.3.1 QwtColorMap::QwtColorMap ([Format](#) = QwtColorMap::RGB)

Constructor.

Definition at line 164 of file qwt_color_map.cpp.

6.9.3.2 QwtColorMap::~QwtColorMap () [virtual]

Destructor.

Definition at line 170 of file qwt_color_map.cpp.

6.9.4 Member Function Documentation

6.9.4.1 [QwtColorMap::Format](#) QwtColorMap::format () const [inline]

Returns:

Intended format of the color map

See also:

[Format](#)

Definition at line 216 of file qwt_color_map.h.

Referenced by QwtPainter::drawColorBar().

6.9.4.2 virtual [QwtColorMap*](#) QwtColorMap::copy () const [pure virtual]

Clone the color map.

Implemented in [QwtLinearColorMap](#), and [QwtAlphaColorMap](#).

Referenced by QwtScaleWidget::setColorMap(), and QwtPlotSpectrogram::setColorMap().

6.9.4.3 virtual [QRgb](#) QwtColorMap::rgb (const [QwtDoubleInterval](#) & *interval*, double *value*) const [pure virtual]

Map a value of a given interval into a rgb value.

Parameters:

interval Range for the values

value Value

Returns:

rgb value, corresponding to value

Implemented in [QwtLinearColorMap](#), and [QwtAlphaColorMap](#).

Referenced by color(), colorTable(), and QwtPainter::drawColorBar().

6.9.4.4 `virtual unsigned char QwtColorMap::colorIndex (const QwtDoubleInterval & interval, double value) const` `[pure virtual]`

Map a value of a given interval into a color index

Parameters:

interval Range for the values

value Value

Returns:

color index, corresponding to value

Implemented in [QwtLinearColorMap](#).

Referenced by `color()`, and `QwtPainter::drawColorBar()`.

6.9.4.5 `QColor QwtColorMap::color (const QwtDoubleInterval & interval, double value) const` `[inline]`

Map a value into a color

Parameters:

interval Valid interval for values

value Value

Returns:

Color corresponding to value

Warning:

This method is slow for Indexed color maps. If it is necessary to map many values, its better to get the color table once and find the color using [colorIndex\(\)](#).

Definition at line 198 of file `qwt_color_map.h`.

References `colorIndex()`, `colorTable()`, and `rgb()`.

6.9.4.6 `QwtColorTable QwtColorMap::colorTable (const QwtDoubleInterval & interval) const` `[virtual]`

Build and return a color map of 256 colors

The color table is needed for rendering indexed images in combination with using [colorIndex\(\)](#).

Parameters:

interval Range for the values

Returns:

A color table, that can be used for a `QImage`

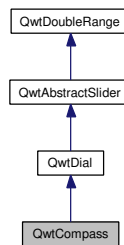
Definition at line 183 of file `qwt_color_map.cpp`.

References `QwtDoubleInterval::isValid()`, `QwtDoubleInterval::minValue()`, `rgb()`, and `QwtDoubleInterval::width()`.

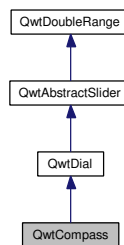
Referenced by `color()`, and `QwtPainter::drawColorBar()`.

6.10 QwtCompass Class Reference

Inheritance diagram for QwtCompass:



Collaboration diagram for QwtCompass:



6.10.1 Detailed Description

A Compass Widget.

[QwtCompass](#) is a widget to display and enter directions. It consists of a scale, an optional needle and rose.

Note:

The examples/dials example shows how to use [QwtCompass](#).

Definition at line 48 of file `qwt_compass.h`.

Public Member Functions

- [QwtCompass](#) (QWidget *parent=NULL)
- virtual [~QwtCompass](#) ()
- void [setRose](#) ([QwtCompassRose](#) *rose)
- const [QwtCompassRose](#) * [rose](#) () const
- [QwtCompassRose](#) * [rose](#) ()
- const QMap< double, QString > & [labelMap](#) () const
- QMap< double, QString > & [labelMap](#) ()
- void [setLabelMap](#) (const QMap< double, QString > &map)

Protected Member Functions

- virtual [QwtText scaleLabel](#) (double value) const
- virtual void [drawRose](#) (QPainter *, const QPoint ¢er, int radius, double north, QPalette::ColorGroup) const
- virtual void [drawScaleContents](#) (QPainter *, const QPoint ¢er, int radius) const
- virtual void [keyPressEvent](#) (QKeyEvent *)

6.10.2 Constructor & Destructor Documentation

6.10.2.1 QwtCompass::QwtCompass (QWidget * *parent* = NULL) [explicit]

Constructor.

Parameters:

parent Parent widget

Create a compass widget with a scale, no needle and no rose. The default origin is 270.0 with no valid value. It accepts mouse and keyboard inputs and has no step size. The default mode is QwtDial::RotateNeedle.

Definition at line 50 of file qwt_compass.cpp.

6.10.2.2 QwtCompass::~QwtCompass () [virtual]

Destructor.

Definition at line 77 of file qwt_compass.cpp.

6.10.3 Member Function Documentation

6.10.3.1 void QwtCompass::setRose (QwtCompassRose * *rose*)

Set a rose for the compass

Parameters:

rose Compass rose

Warning:

The rose will be deleted, when a different rose is set or in ~QwtCompass

See also:

[rose\(\)](#)

Definition at line 157 of file qwt_compass.cpp.

References [rose\(\)](#).

6.10.3.2 const QwtCompassRose * QwtCompass::rose () const

Returns:

rose

See also:

[setRose\(\)](#)

Definition at line 173 of file qwt_compass.cpp.

Referenced by [setRose\(\)](#).

6.10.3.3 [QwtCompassRose](#) * [QwtCompass::rose \(\)](#)

Returns:

rose

See also:

[setRose\(\)](#)

Definition at line 182 of file qwt_compass.cpp.

6.10.3.4 `const QMap< double, QString > & QwtCompass::labelMap () const`

Returns:

map, mapping values to labels

See also:

[setLabelMap\(\)](#)

Definition at line 256 of file qwt_compass.cpp.

6.10.3.5 `QMap< double, QString > & QwtCompass::labelMap ()`

Returns:

map, mapping values to labels

See also:

[setLabelMap\(\)](#)

Definition at line 265 of file qwt_compass.cpp.

6.10.3.6 `void QwtCompass::setLabelMap (const QMap< double, QString > & map)`

Set a map, mapping values to labels.

Parameters:

map value to label map

The values of the major ticks are found by looking into this map. The default map consists of the labels N, NE, E, SE, S, SW, W, NW.

Warning:

The map will have no effect for values that are no major tick values. Major ticks can be changed by `QwtScaleDraw::setScale`

See also:

[labelMap\(\)](#), [scaleDraw\(\)](#), [setScale\(\)](#)

Definition at line 282 of file `qwt_compass.cpp`.

6.10.3.7 `QwtText` `QwtCompass::scaleLabel (double value) const` [protected, virtual]

Map a value to a corresponding label

Parameters:

value Value that will be mapped

Returns:

Label, or `QString::null`

`label()` looks in a map for a corresponding label for value or return an null text.

See also:

[labelMap\(\)](#), [setLabelMap\(\)](#)

Reimplemented from [QwtDial](#).

Definition at line 297 of file `qwt_compass.cpp`.

6.10.3.8 `void QwtCompass::drawRose (QPainter * painter, const QPoint & center, int radius, double north, QPalette::ColorGroup cg) const` [protected, virtual]

Draw the compass rose

Parameters:

painter Painter

center Center of the compass

radius of the circle, where to paint the rose

north Direction pointing north, in degrees counter clockwise

cg Color group

Definition at line 143 of file `qwt_compass.cpp`.

Referenced by `drawScaleContents()`.

6.10.3.9 `void QwtCompass::drawScaleContents (QPainter *, const QPoint & center, int radius) const` [protected, virtual]

Draw the contents of the scale.

Reimplemented from [QwtDial](#).

Definition at line 114 of file qwt_compass.cpp.

References [drawRose\(\)](#), [QwtAbstractSlider::isValid\(\)](#), [QwtDial::mode\(\)](#), [QwtDial::origin\(\)](#), and [QwtDoubleRange::value\(\)](#).

6.10.3.10 void QwtCompass::keyPressEvent (QKeyEvent * *kev*) [protected, virtual]

Handles key events

Beside the keys described in [QwtDial::keyPressEvent](#) numbers from 1-9 (without 5) set the direction according to their position on the num pad.

See also:

[isReadOnly\(\)](#)

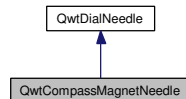
Reimplemented from [QwtDial](#).

Definition at line 196 of file qwt_compass.cpp.

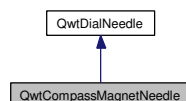
References [QwtAbstractSlider::isReadOnly\(\)](#), [QwtDial::keyPressEvent\(\)](#), [QwtDial::mode\(\)](#), [QwtDial::origin\(\)](#), [QwtAbstractSlider::setValue\(\)](#), and [QwtDoubleRange::value\(\)](#).

6.11 QwtCompassMagnetNeedle Class Reference

Inheritance diagram for QwtCompassMagnetNeedle:



Collaboration diagram for QwtCompassMagnetNeedle:



6.11.1 Detailed Description

A magnet needle for compass widgets.

A magnet needle points to two opposite directions indicating north and south.

The following colors are used:

- [QColorGroup::Light](#)
Used for pointing south
- [QColorGroup::Dark](#)
Used for pointing north

- QColorGroup::Base
Knob (ThinStyle only)

See also:

[QwtDial](#), [QwtCompass](#)

Definition at line 125 of file qwt_dial_needle.h.

Public Types

- enum [Style](#) {
 Arrow,
 Ray,
 TriangleStyle,
 ThinStyle,
 Style1,
 Style2,
 NoSymbol = -1,
 Ellipse,
 Rect,
 Diamond,
 Triangle,
 DTriangle,
 UTriangle,
 LTriangle,
 RTriangle,
 Cross,
 XCross,
 HLine,
 VLine,
 Star1,
 Star2,
 Hexagon,
 StyleCnt }

Public Member Functions

- [QwtCompassMagnetNeedle](#) ([Style](#)=TriangleStyle, const QColor &light=Qt::white, const QColor &dark=Qt::red)
- virtual void [draw](#) (QPainter *, const QPoint &, int length, double direction, QPalette::ColorGroup=QPalette::Active) const

Static Public Member Functions

- static void [drawTriangleNeedle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, double direction)
- static void [drawThinNeedle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, double direction)

Static Protected Member Functions

- static void [drawPointer](#) (QPainter *painter, const QBrush &brush, int colorOffset, const QPoint ¢er, int length, int width, double direction)

6.11.2 Member Enumeration Documentation

6.11.2.1 enum [QwtCompassMagnetNeedle::Style](#)

Style of the needle.

Definition at line 129 of file qwt_dial_needle.h.

6.11.3 Constructor & Destructor Documentation

6.11.3.1 [QwtCompassMagnetNeedle::QwtCompassMagnetNeedle](#) ([Style](#) = TriangleStyle, const QColor & *light* = Qt::white, const QColor & *dark* = Qt::red)

Constructor.

Definition at line 265 of file qwt_dial_needle.cpp.

References [QwtDialNeedle::palette\(\)](#), and [QwtDialNeedle::setPalette\(\)](#).

6.11.4 Member Function Documentation

6.11.4.1 void [QwtCompassMagnetNeedle::draw](#) (QPainter * *painter*, const QPoint & *center*, int *length*, double *direction*, QPalette::ColorGroup *colorGroup* = QPalette::Active) const [virtual]

Draw the needle

Parameters:

painter Painter

center Center of the dial, start position for the needle

length Length of the needle

direction Direction of the needle, in degrees counter clockwise

colorGroup Color group, used for painting

Implements [QwtDialNeedle](#).

Definition at line 292 of file qwt_dial_needle.cpp.

References [drawThinNeedle\(\)](#), [drawTriangleNeedle\(\)](#), and [QwtDialNeedle::palette\(\)](#).

6.11.4.2 void QwtCompassMagnetNeedle::drawTriangleNeedle (QPainter * *painter*, const QPalette & *palette*, QPalette::ColorGroup *colorGroup*, const QPoint & *center*, int *length*, double *direction*) [static]

Draw a compass needle

Parameters:

painter Painter

palette Palette

colorGroup Color group

center Center, where the needle starts

length Length of the needle

direction Direction

Definition at line 317 of file qwt_dial_needle.cpp.

Referenced by draw().

6.11.4.3 void QwtCompassMagnetNeedle::drawThinNeedle (QPainter * *painter*, const QPalette & *palette*, QPalette::ColorGroup *colorGroup*, const QPoint & *center*, int *length*, double *direction*) [static]

Draw a compass needle

Parameters:

painter Painter

palette Palette

colorGroup Color group

center Center, where the needle starts

length Length of the needle

direction Direction

Definition at line 383 of file qwt_dial_needle.cpp.

References QwtDialNeedle::drawKnob(), and drawPointer().

Referenced by draw().

6.11.4.4 void QwtCompassMagnetNeedle::drawPointer (QPainter * *painter*, const QBrush & *brush*, int *colorOffset*, const QPoint & *center*, int *length*, int *width*, double *direction*) [static, protected]

Draw a compass needle

Parameters:

painter Painter

brush Brush

colorOffset Color offset

center Center, where the needle starts

length Length of the needle

width Width of the needle

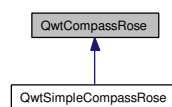
direction Direction

Definition at line 419 of file qwt_dial_needle.cpp.

Referenced by drawThinNeedle().

6.12 QwtCompassRose Class Reference

Inheritance diagram for QwtCompassRose:



6.12.1 Detailed Description

Abstract base class for a compass rose.

Definition at line 21 of file qwt_compass_rose.h.

Public Member Functions

- virtual [~QwtCompassRose](#) ()
- virtual void [setPalette](#) (const QPalette &p)
- const QPalette & [palette](#) () const
- virtual void [draw](#) (QPainter *painter, const QPoint ¢er, int radius, double north, QPalette::ColorGroup colorGroup=QPalette::Active) const=0

6.12.2 Member Function Documentation

6.12.2.1 virtual void [QwtCompassRose::draw](#) (QPainter * *painter*, const QPoint & *center*, int *radius*, double *north*, QPalette::ColorGroup *colorGroup* = QPalette::Active) const [pure virtual]

Draw the rose

Parameters:

painter Painter

center Center point

radius Radius of the rose

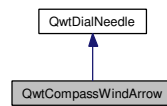
north Position

colorGroup Color group

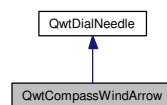
Implemented in [QwtSimpleCompassRose](#).

6.13 QwtCompassWindArrow Class Reference

Inheritance diagram for QwtCompassWindArrow:



Collaboration diagram for QwtCompassWindArrow:



6.13.1 Detailed Description

An indicator for the wind direction.

[QwtCompassWindArrow](#) shows the direction where the wind comes from.

- `QColorGroup::Light`
Used for Style1, or the light half of Style2
- `QColorGroup::Dark`
Used for the dark half of Style2

See also:

[QwtDial](#), [QwtCompass](#)

Definition at line 170 of file `qwt_dial_needle.h`.

Public Types

- enum [Style](#) {
 Arrow,
 Ray,
 TriangleStyle,
 ThinStyle,
 Style1,
 Style2,
 NoSymbol = -1,
 Ellipse,
 Rect,
 Diamond,

Triangle,
 DTriangle,
 UTriangle,
 LTriangle,
 RTriangle,
 Cross,
 XCross,
 HLine,
 VLine,
 Star1,
 Star2,
 Hexagon,
 StyleCnt }

Public Member Functions

- [QwtCompassWindArrow](#) ([Style](#), const QColor &light=Qt::white, const QColor &dark=Qt::gray)
- virtual void [draw](#) (QPainter *, const QPoint &, int length, double direction, QPalette::ColorGroup=QPalette::Active) const

Static Public Member Functions

- static void [drawStyle1Needle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, double direction)
- static void [drawStyle2Needle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, double direction)

6.13.2 Member Enumeration Documentation

6.13.2.1 enum [QwtCompassWindArrow::Style](#)

Style of the arrow.

Definition at line 174 of file qwt_dial_needle.h.

6.13.3 Constructor & Destructor Documentation

6.13.3.1 [QwtCompassWindArrow::QwtCompassWindArrow](#) ([Style](#) *style*, const QColor & *light* = Qt::white, const QColor & *dark* = Qt::gray)

Constructor

Parameters:

style Arrow style
light Light color
dark Dark color

Definition at line 467 of file qwt_dial_needle.cpp.

References [QwtDialNeedle::palette\(\)](#), and [QwtDialNeedle::setPalette\(\)](#).

6.13.4 Member Function Documentation

6.13.4.1 `void QwtCompassWindArrow::draw (QPainter * painter, const QPoint & center, int length, double direction, QPalette::ColorGroup colorGroup = QPalette::Active) const`
[virtual]

Draw the needle

Parameters:

painter Painter

center Center of the dial, start position for the needle

length Length of the needle

direction Direction of the needle, in degrees counter clockwise

colorGroup Color group, used for painting

Implements [QwtDialNeedle](#).

Definition at line 492 of file `qwt_dial_needle.cpp`.

References `drawStyle1Needle()`, `drawStyle2Needle()`, and `QwtDialNeedle::palette()`.

6.13.4.2 `void QwtCompassWindArrow::drawStyle1Needle (QPainter * painter, const QPalette & palette, QPalette::ColorGroup colorGroup, const QPoint & center, int length, double direction)`
[static]

Draw a compass needle

Parameters:

painter Painter

palette Palette

colorGroup colorGroup

center Center of the dial, start position for the needle

length Length of the needle

direction Direction of the needle, in degrees counter clockwise

Definition at line 517 of file `qwt_dial_needle.cpp`.

Referenced by `draw()`.

6.13.4.3 `void QwtCompassWindArrow::drawStyle2Needle (QPainter * painter, const QPalette & palette, QPalette::ColorGroup colorGroup, const QPoint & center, int length, double direction)`
[static]

Draw a compass needle

Parameters:

painter Painter

palette Palette

colorGroup colorGroup

center Center of the dial, start position for the needle

length Length of the needle

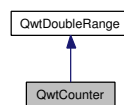
direction Direction of the needle, in degrees counter clockwise

Definition at line 554 of file qwt_dial_needle.cpp.

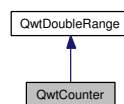
Referenced by draw().

6.14 QwtCounter Class Reference

Inheritance diagram for QwtCounter:



Collaboration diagram for QwtCounter:



6.14.1 Detailed Description

The Counter Widget.

A Counter consists of a label displaying a number and one ore more (up to three) push buttons on each side of the label which can be used to increment or decrement the counter's value.

A Counter has a range from a minimum value to a maximum value and a step size. The range can be specified using `QwtDblRange::setRange()`. The counter's value is an integer multiple of the step size. The number of steps by which a button increments or decrements the value can be specified using `QwtCounter::setIncSteps()`. The number of buttons can be changed with `QwtCounter::setNumButtons()`.

Holding the space bar down with focus on a button is the fastest method to step through the counter values. When the counter underflows/overflows, the focus is set to the smallest up/down button and counting is disabled. Counting is re-enabled on a button release event (mouse or space bar).

Example:

```

#include "../include/qwt_counter.h>

QwtCounter *cnt;

cnt = new QwtCounter(parent, name);

cnt->setRange(0.0, 100.0, 1.0);           // From 0.0 to 100, step 1.0
cnt->setNumButtons(2);                   // Two buttons each side
cnt->setIncSteps(QwtCounter::Button1, 1); // Button 1 increments 1 step
cnt->setIncSteps(QwtCounter::Button2, 20); // Button 2 increments 20 steps

connect(cnt, SIGNAL(valueChanged(double)), my_class, SLOT(newValue(double)));
  
```

Definition at line 60 of file qwt_counter.h.

Public Types

- enum [Button](#) {
 [Button1](#),
 [Button2](#),
 [Button3](#),
 [ButtonCnt](#) }

Signals

- void [buttonReleased](#) (double value)
- void [valueChanged](#) (double value)

Public Member Functions

- [QwtCounter](#) (QWidget *parent=NULL)
- virtual [~QwtCounter](#) ()
- bool [editable](#) () const
- void [setEditable](#) (bool)
- void [setNumButtons](#) (int n)
- int [numButtons](#) () const
- void [setIncSteps](#) ([QwtCounter::Button](#) btn, int nSteps)
- int [incSteps](#) ([QwtCounter::Button](#) btn) const
- virtual void [setValue](#) (double)
- virtual QSize [sizeHint](#) () const
- virtual void [polish](#) ()
- double [step](#) () const
- void [setStep](#) (double s)
- double [minVal](#) () const
- void [setMinValue](#) (double m)
- double [maxVal](#) () const
- void [setMaxValue](#) (double m)
- void [setStepButton1](#) (int nSteps)
- int [stepButton1](#) () const
- void [setStepButton2](#) (int nSteps)
- int [stepButton2](#) () const
- void [setStepButton3](#) (int nSteps)
- int [stepButton3](#) () const
- virtual double [value](#) () const

Protected Member Functions

- virtual bool [event](#) (QEvent *)
- virtual void [wheelEvent](#) (QWheelEvent *)
- virtual void [keyPressEvent](#) (QKeyEvent *)
- virtual void [rangeChange](#) ()

6.14.2 Member Enumeration Documentation

6.14.2.1 enum [QwtCounter::Button](#)

Button index

Definition at line 79 of file qwt_counter.h.

6.14.3 Constructor & Destructor Documentation

6.14.3.1 [QwtCounter::QwtCounter \(QWidget **parent* = NULL\)](#) [explicit]

The default number of buttons is set to 2. The default increments are:

- Button 1: 1 step
- Button 2: 10 steps
- Button 3: 100 steps

Parameters:

parent

Definition at line 50 of file qwt_counter.cpp.

6.14.3.2 [QwtCounter::~~QwtCounter \(\)](#) [virtual]

Destructor.

Definition at line 143 of file qwt_counter.cpp.

6.14.4 Member Function Documentation

6.14.4.1 [bool QwtCounter::editable \(\) const](#)

returns whether the line edit is edatble. (default is yes)

Definition at line 198 of file qwt_counter.cpp.

6.14.4.2 [void QwtCounter::setEditable \(bool *editable*\)](#)

Allow/disallow the user to manually edit the value.

Parameters:

editable true enables editing

See also:

[editable\(\)](#)

Definition at line 185 of file qwt_counter.cpp.

6.14.4.3 void QwtCounter::setNumButtons (int *n*)

Specify the number of buttons on each side of the label.

Parameters:

n Number of buttons

Definition at line 441 of file qwt_counter.cpp.

6.14.4.4 int QwtCounter::numButtons () const**Returns:**

The number of buttons on each side of the widget.

Definition at line 466 of file qwt_counter.cpp.

6.14.4.5 void QwtCounter::setIncSteps (QwtCounter::Button *btn*, int *nSteps*)

Specify the number of steps by which the value is incremented or decremented when a specified button is pushed.

Parameters:

btn One of QwtCounter::Button1, QwtCounter::Button2, QwtCounter::Button3
nSteps Number of steps

Definition at line 355 of file qwt_counter.cpp.

Referenced by setStepButton1(), setStepButton2(), and setStepButton3().

6.14.4.6 int QwtCounter::incSteps (QwtCounter::Button *btn*) const**Returns:**

the number of steps by which a specified button increments the value or 0 if the button is invalid.

Parameters:

btn One of QwtCounter::Button1, QwtCounter::Button2, QwtCounter::Button3

Definition at line 367 of file qwt_counter.cpp.

Referenced by stepButton1(), stepButton2(), and stepButton3().

6.14.4.7 void QwtCounter::setValue (double *v*) [virtual]

Set a new value.

Parameters:

v new value Calls QwtDoubleRange::setValue and does all visual updates.

See also:

[QwtDoubleRange::setValue](#)

Reimplemented from [QwtDoubleRange](#).

Definition at line 382 of file qwt_counter.cpp.

References [QwtDoubleRange::setValue\(\)](#), and [value\(\)](#).

Referenced by [keyPressEvent\(\)](#).

6.14.4.8 QSize QwtCounter::sizeHint () const [virtual]

A size hint.

Definition at line 513 of file qwt_counter.cpp.

References [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), and [step\(\)](#).

6.14.4.9 void QwtCounter::polish () [virtual]

Sets the minimum width for the buttons

Definition at line 151 of file qwt_counter.cpp.

Referenced by [event\(\)](#).

6.14.4.10 double QwtCounter::step () const

returns the step size

Reimplemented from [QwtDoubleRange](#).

Definition at line 550 of file qwt_counter.cpp.

References [QwtDoubleRange::step\(\)](#).

Referenced by [setMaxValue\(\)](#), [setMinValue\(\)](#), and [sizeHint\(\)](#).

6.14.4.11 void QwtCounter::setStep (double s)

sets the step size

Reimplemented from [QwtDoubleRange](#).

Definition at line 556 of file qwt_counter.cpp.

References [QwtDoubleRange::setStep\(\)](#).

6.14.4.12 double QwtCounter::minVal () const

returns the minimum value of the range

Definition at line 562 of file qwt_counter.cpp.

References [QwtDoubleRange::minValue\(\)](#).

6.14.4.13 void QwtCounter::setMinValue (double m)

sets the minimum value of the range

Definition at line 568 of file qwt_counter.cpp.

References QwtDoubleRange::maxValue(), QwtDoubleRange::setRange(), and step().

6.14.4.14 double QwtCounter::maxVal () const

returns the maximum value of the range

Definition at line 574 of file qwt_counter.cpp.

References QwtDoubleRange::maxValue().

6.14.4.15 void QwtCounter::setMaxValue (double *m*)

sets the maximum value of the range

Definition at line 580 of file qwt_counter.cpp.

References QwtDoubleRange::minValue(), QwtDoubleRange::setRange(), and step().

6.14.4.16 void QwtCounter::setStepButton1 (int *nSteps*)

set the number of increment steps for button 1

Definition at line 586 of file qwt_counter.cpp.

References setIncSteps().

6.14.4.17 int QwtCounter::stepButton1 () const

returns the number of increment steps for button 1

Definition at line 592 of file qwt_counter.cpp.

References incSteps().

6.14.4.18 void QwtCounter::setStepButton2 (int *nSteps*)

set the number of increment steps for button 2

Definition at line 598 of file qwt_counter.cpp.

References setIncSteps().

6.14.4.19 int QwtCounter::stepButton2 () const

returns the number of increment steps for button 2

Definition at line 604 of file qwt_counter.cpp.

References incSteps().

6.14.4.20 void QwtCounter::setStepButton3 (int *nSteps*)

set the number of increment steps for button 3

Definition at line 610 of file qwt_counter.cpp.

References setIncSteps().

6.14.4.21 int QwtCounter::stepButton3 () const

returns the number of increment steps for button 3

Definition at line 616 of file qwt_counter.cpp.

References incSteps().

6.14.4.22 double QwtCounter::value () const [virtual]

Returns the current value.

Reimplemented from [QwtDoubleRange](#).

Definition at line 621 of file qwt_counter.cpp.

References QwtDoubleRange::value().

Referenced by setValue().

6.14.4.23 void QwtCounter::buttonReleased (double *value*) [signal]

This signal is emitted when a button has been released

Parameters:

value The new value

6.14.4.24 void QwtCounter::valueChanged (double *value*) [signal]

This signal is emitted when the counter's value has changed

Parameters:

value The new value

6.14.4.25 bool QwtCounter::event (QEvent * *e*) [protected, virtual]

Handle PolishRequest events

Definition at line 206 of file qwt_counter.cpp.

References polish().

6.14.4.26 void QwtCounter::keyPressEvent (QKeyEvent * *e*) [protected, virtual]

Handles key events

- Ctrl + Qt::Key_Home Step to [minValue\(\)](#)
- Ctrl + Qt::Key_End Step to [maxValue\(\)](#)
- Qt::Key_Up Increment by incSteps(QwtCounter::Button1)
- Qt::Key_Down Decrement by incSteps(QwtCounter::Button1)
- Qt::Key_PageUp Increment by incSteps(QwtCounter::Button2)
- Qt::Key_PageDown Decrement by incSteps(QwtCounter::Button2)

- Shift + Qt::Key_PageUp Increment by incSteps(QwtCounter::Button3)
- Shift + Qt::Key_PageDown Decrement by incSteps(QwtCounter::Button3)

Definition at line 236 of file qwt_counter.cpp.

References [QwtDoubleRange::incValue\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), and [setValue\(\)](#).

6.14.4.27 void QwtCounter::rangeChange () [protected, virtual]

Notify change of range.

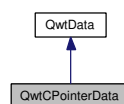
This function updates the enabled property of all buttons contained in [QwtCounter](#).

Reimplemented from [QwtDoubleRange](#).

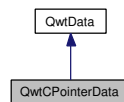
Definition at line 507 of file qwt_counter.cpp.

6.15 QwtCPointerData Class Reference

Inheritance diagram for QwtCPointerData:



Collaboration diagram for QwtCPointerData:



6.15.1 Detailed Description

Data class containing two pointers to memory blocks of doubles.

Definition at line 144 of file qwt_data.h.

Public Member Functions

- [QwtCPointerData](#) (const double *x, const double *y, size_t size)
- [QwtCPointerData](#) & operator= (const [QwtCPointerData](#) &)
- virtual [QwtData](#) * copy () const
- virtual size_t size () const
- virtual double x (size_t i) const
- virtual double y (size_t i) const
- const double * xData () const
- const double * yData () const
- virtual [QwtDoubleRect](#) boundingRect () const

6.15.2 Constructor & Destructor Documentation

6.15.2.1 QwtCPointerData::QwtCPointerData (const double * x, const double * y, size_t size)

Constructor

Parameters:

- x* Array of x values
- y* Array of y values
- size* Size of the x and y arrays

Warning:

The programmer must assure that the memory blocks referenced by the pointers remain valid during the lifetime of the QwtPlotCPointer object.

See also:

[QwtPlotCurve::setData\(\)](#), [QwtPlotCurve::setRawData\(\)](#)

Definition at line 281 of file qwt_data.cpp.

Referenced by [copy\(\)](#).

6.15.3 Member Function Documentation

6.15.3.1 QwtCPointerData & QwtCPointerData::operator= (const QwtCPointerData &)

Assignment.

Definition at line 290 of file qwt_data.cpp.

References [d_size](#), [d_x](#), and [d_y](#).

6.15.3.2 QwtData * QwtCPointerData::copy () const [virtual]

Returns:

Pointer to a copy (virtual copy constructor)

Implements [QwtData](#).

Definition at line 344 of file qwt_data.cpp.

References [QwtCPointerData\(\)](#).

6.15.3.3 size_t QwtCPointerData::size () const [virtual]

Returns:

Size of the data set

Implements [QwtData](#).

Definition at line 302 of file qwt_data.cpp.

Referenced by [boundingRect\(\)](#).

6.15.3.4 double QwtCPointerData::x (size_t i) const [virtual]

Return the x value of data point i

Parameters:

i Index

Returns:

x X value of data point i

Implements [QwtData](#).

Definition at line 313 of file qwt_data.cpp.

6.15.3.5 double QwtCPointerData::y (size_t i) const [virtual]

Return the y value of data point i

Parameters:

i Index

Returns:

y Y value of data point i

Implements [QwtData](#).

Definition at line 324 of file qwt_data.cpp.

6.15.3.6 const double * QwtCPointerData::xData () const**Returns:**

Array of the x-values

Definition at line 330 of file qwt_data.cpp.

6.15.3.7 const double * QwtCPointerData::yData () const**Returns:**

Array of the y-values

Definition at line 336 of file qwt_data.cpp.

6.15.3.8 [QwtDoubleRect](#) QwtCPointerData::boundingRect () const [virtual]

Returns the bounding rectangle of the data. If there is no bounding rect, like for empty data the rectangle is invalid: [QwtDoubleRect::isValid\(\)](#) == false

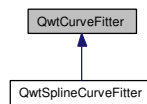
Reimplemented from [QwtData](#).

Definition at line 354 of file qwt_data.cpp.

References [size\(\)](#).

6.16 QwtCurveFitter Class Reference

Inheritance diagram for QwtCurveFitter:



6.16.1 Detailed Description

Abstract base class for a curve fitter.

Definition at line 42 of file qwt_curve_fitter.h.

Public Member Functions

- virtual [~QwtCurveFitter](#) ()
- virtual QPolygonF **fitCurve** (const QPolygonF &) const=0

Protected Member Functions

- [QwtCurveFitter](#) ()

6.16.2 Constructor & Destructor Documentation

6.16.2.1 QwtCurveFitter::~~QwtCurveFitter () [virtual]

Destructor.

Definition at line 20 of file qwt_curve_fitter.cpp.

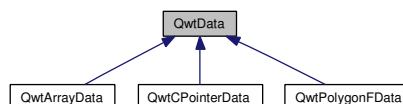
6.16.2.2 QwtCurveFitter::QwtCurveFitter () [protected]

Constructor.

Definition at line 15 of file qwt_curve_fitter.cpp.

6.17 QwtData Class Reference

Inheritance diagram for QwtData:



6.17.1 Detailed Description

[QwtData](#) defines an interface to any type of curve data.

Classes, derived from [QwtData](#) may:

- store the data in almost any type of container
- calculate the data on the fly instead of storing it

Definition at line 47 of file `qwt_data.h`.

Public Member Functions

- [QwtData](#) ()
- virtual [~QwtData](#) ()
- virtual [QwtData](#) * [copy](#) () const=0
- virtual `size_t` [size](#) () const=0
- virtual `double` [x](#) (`size_t` i) const=0
- virtual `double` [y](#) (`size_t` i) const=0
- virtual [QwtDoubleRect](#) [boundingRect](#) () const

Protected Member Functions

- [QwtData](#) & [operator=](#) (const [QwtData](#) &)

6.17.2 Constructor & Destructor Documentation

6.17.2.1 [QwtData::QwtData](#) ()

Constructor.

Definition at line 14 of file `qwt_data.cpp`.

6.17.2.2 [QwtData::~~QwtData](#) () [virtual]

Destructor.

Definition at line 19 of file `qwt_data.cpp`.

6.17.3 Member Function Documentation

6.17.3.1 virtual [QwtData](#)* [QwtData::copy](#) () const [pure virtual]

Returns:

Pointer to a copy (virtual copy constructor)

Implemented in [QwtPolygonFData](#), [QwtArrayData](#), and [QwtCPointerData](#).

Referenced by `QwtPlotCurve::setData()`.

6.17.3.2 virtual size_t QwtData::size () const [pure virtual]**Returns:**

Size of the data set

Implemented in [QwtPolygonFData](#), [QwtArrayData](#), and [QwtCPointerData](#).

Referenced by [boundingRect\(\)](#), and [QwtPlotCurve::dataSize\(\)](#).

6.17.3.3 virtual double QwtData::x (size_t i) const [pure virtual]

Return the x value of data point i

Parameters:

i Index

Returns:

x X value of data point i

Implemented in [QwtPolygonFData](#), [QwtArrayData](#), and [QwtCPointerData](#).

Referenced by [boundingRect\(\)](#), and [QwtPlotCurve::x\(\)](#).

6.17.3.4 virtual double QwtData::y (size_t i) const [pure virtual]

Return the y value of data point i

Parameters:

i Index

Returns:

y Y value of data point i

Implemented in [QwtPolygonFData](#), [QwtArrayData](#), and [QwtCPointerData](#).

Referenced by [boundingRect\(\)](#), and [QwtPlotCurve::y\(\)](#).

6.17.3.5 QwtDoubleRect QwtData::boundingRect () const [virtual]

Returns the bounding rectangle of the data. If there is no bounding rect, like for empty data the rectangle is invalid: [QwtDoubleRect::isValid\(\)](#) == false

Warning:

This is an slow implementation iterating over all points. It is intended to be overloaded by derived classes. In case of auto scaling [boundingRect\(\)](#) is called for every replot, so it might be worth to implement a cache, or use [x\(0\)](#), [x\(size\(\) - 1\)](#) for ordered data ...

Reimplemented in [QwtArrayData](#), and [QwtCPointerData](#).

Definition at line 34 of file [qwt_data.cpp](#).

References [size\(\)](#), [x\(\)](#), and [y\(\)](#).

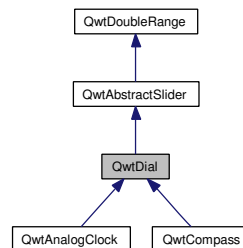
Referenced by [QwtPlotCurve::boundingRect\(\)](#).

6.17.3.6 [QwtData&](#) [QwtData::operator=](#) (const [QwtData](#) &) [protected]

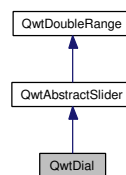
Assignment operator (virtualized)

6.18 QwtDial Class Reference

Inheritance diagram for QwtDial:



Collaboration diagram for QwtDial:



6.18.1 Detailed Description

[QwtDial](#) class provides a rounded range control.

[QwtDial](#) is intended as base class for dial widgets like speedometers, compass widgets, clocks ...

A dial contains a scale and a needle indicating the current value of the dial. Depending on Mode one of them is fixed and the other is rotating. If not [isReadOnly\(\)](#) the dial can be rotated by dragging the mouse or using keyboard inputs (see [keyPressEvent\(\)](#)). A dial might be wrapping, what means a rotation below/above one limit continues on the other limit (f.e compass). The scale might cover any arc of the dial, its values are related to the [origin\(\)](#) of the dial.

Qwt is missing a set of good looking needles ([QwtDialNeedle](#)). Contributions are very welcome.

See also:

[QwtCompass](#), [QwtAnalogClock](#), [QwtDialNeedle](#)

Note:

The examples/dials example shows different types of dials.

Definition at line 67 of file `qwt_dial.h`.

Public Types

- enum [Shadow](#) {

```

Plain = QFrame::Plain,
Raised = QFrame::Raised,
Sunken = QFrame::Sunken }
• enum ScaleOptions {
    ScaleBackbone = 1,
    ScaleTicks = 2,
    ScaleLabel = 4 }
• enum Mode {
    FixedColors,
    ScaledColors,
    RotateNeedle,
    RotateScale }

```

Public Member Functions

- [QwtDial](#) (QWidget *parent=NULL)
- virtual [~QwtDial](#) ()
- void [setFrameShadow](#) ([Shadow](#))
- [Shadow](#) [frameShadow](#) () const
- bool [hasVisibleBackground](#) () const
- void [showBackground](#) (bool)
- void [setLineWidth](#) (int)
- int [lineWidth](#) () const
- void [setMode](#) ([Mode](#))
- [Mode](#) [mode](#) () const
- virtual void [setWrapping](#) (bool)
- bool [wrapping](#) () const
- virtual void [setScale](#) (int maxMajIntv, int maxMinIntv, double step=0.0)
- void [setScaleArc](#) (double min, double max)
- void [setScaleOptions](#) (int)
- void [setScaleTicks](#) (int minLen, int medLen, int majLen, int penWidth=1)
- double [minScaleArc](#) () const
- double [maxScaleArc](#) () const
- virtual void [setOrigin](#) (double)
- double [origin](#) () const
- virtual void [setNeedle](#) ([QwtDialNeedle](#) *)
- const [QwtDialNeedle](#) * [needle](#) () const
- [QwtDialNeedle](#) * [needle](#) ()
- QRect [boundingRect](#) () const
- QRect [contentsRect](#) () const
- virtual QRect [scaleContentsRect](#) () const
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- virtual void [setScaleDraw](#) ([QwtDialScaleDraw](#) *)
- [QwtDialScaleDraw](#) * [scaleDraw](#) ()
- const [QwtDialScaleDraw](#) * [scaleDraw](#) () const

Protected Member Functions

- virtual void [paintEvent](#) (QPaintEvent *)
- virtual void [resizeEvent](#) (QResizeEvent *)
- virtual void [keyPressEvent](#) (QKeyEvent *)
- virtual void [updateMask](#) ()
- virtual void [drawFrame](#) (QPainter *p)
- virtual void [drawContents](#) (QPainter *) const
- virtual void [drawFocusIndicator](#) (QPainter *) const
- virtual void [drawScale](#) (QPainter *, const QPoint ¢er, int radius, double origin, double arcMin, double arcMax) const
- virtual void [drawScaleContents](#) (QPainter *painter, const QPoint ¢er, int radius) const
- virtual void [drawNeedle](#) (QPainter *, const QPoint &, int radius, double direction, QPalette::ColorGroup) const
- virtual [QwtText scaleLabel](#) (double) const
- void [updateScale](#) ()
- virtual void [rangeChange](#) ()
- virtual void [valueChange](#) ()
- virtual double [getValue](#) (const QPoint &)
- virtual void [getScrollMode](#) (const QPoint &, int &scrollMode, int &direction)

Friends

- class [QwtDialScaleDraw](#)

6.18.2 Member Enumeration Documentation

6.18.2.1 enum [QwtDial::Shadow](#)

Frame shadow.

Unfortunately it is not possible to use `QFrame::Shadow` as a property of a widget that is not derived from `QFrame`. The following enum is made for the designer only. It is safe to use `QFrame::Shadow` instead.

Definition at line 92 of file `qwt_dial.h`.

6.18.2.2 enum [QwtDial::ScaleOptions](#)

see [QwtDial::setScaleOptions](#)

Definition at line 100 of file `qwt_dial.h`.

6.18.2.3 enum [QwtDial::Mode](#)

In case of `RotateNeedle` the needle is rotating, in case of `RotateScale`, the needle points to [origin\(\)](#) and the scale is rotating.

Definition at line 112 of file `qwt_dial.h`.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 QwtDial::QwtDial (QWidget **parent* = NULL) [explicit]

Constructor.

Parameters:

parent Parent widget

Create a dial widget with no scale and no needle. The default origin is 90.0 with no valid value. It accepts mouse and keyboard inputs and has no step size. The default mode is QwtDial::RotateNeedle.

Definition at line 129 of file qwt_dial.cpp.

6.18.3.2 QwtDial::~QwtDial () [virtual]

Destructor.

Definition at line 193 of file qwt_dial.cpp.

6.18.4 Member Function Documentation

6.18.4.1 void QwtDial::setFrameShadow (Shadow *shadow*)

Sets the frame shadow value from the frame style.

Parameters:

shadow Frame shadow

See also:

[setLineWidth\(\)](#), [QFrame::setFrameShadow\(\)](#)

Definition at line 230 of file qwt_dial.cpp.

References [lineWidth\(\)](#).

6.18.4.2 QwtDial::Shadow QwtDial::frameShadow () const

Returns:

Frame shadow /sa [setFrameShadow\(\)](#), [lineWidth\(\)](#), [QFrame::frameShadow](#)

Definition at line 244 of file qwt_dial.cpp.

6.18.4.3 bool QwtDial::hasVisibleBackground () const

true when the area outside of the frame is visible

See also:

[showBackground\(\)](#), [setMask\(\)](#)

Definition at line 220 of file qwt_dial.cpp.

Referenced by [resizeEvent\(\)](#).

6.18.4.4 void QwtDial::showBackground (bool *show*)

Show/Hide the area outside of the frame

Parameters:

show Show if true, hide if false

See also:

[hasVisibleBackground\(\)](#), [setMask\(\)](#)

Warning:

When [QwtDial](#) is a toplevel widget the window border might disappear too.

Definition at line 206 of file `qwt_dial.cpp`.

References [updateMask\(\)](#).

6.18.4.5 void QwtDial::setLineWidth (int *lineWidth*)

Sets the line width

Parameters:

lineWidth Line width

See also:

[setFrameShadow\(\)](#)

Definition at line 255 of file `qwt_dial.cpp`.

6.18.4.6 int QwtDial::lineWidth () const

Returns:

Line width of the frame

See also:

[setLineWidth\(\)](#), [frameShadow\(\)](#), [lineWidth\(\)](#)

Definition at line 271 of file `qwt_dial.cpp`.

Referenced by [contentsRect\(\)](#), [drawFrame\(\)](#), [minimumSizeHint\(\)](#), [setFrameShadow\(\)](#), and [sizeHint\(\)](#).

6.18.4.7 void QwtDial::setMode ([Mode](#) *mode*)

Change the mode of the meter.

Parameters:

mode New mode

The value of the meter is indicated by the difference between north of the scale and the direction of the needle. In case of `QwtDial::RotateNeedle` north is pointing to the `origin()` and the needle is rotating, in case of `QwtDial::RotateScale`, the needle points to `origin()` and the scale is rotating.

The default mode is `QwtDial::RotateNeedle`.

See also:

`mode()`, `setValue()`, `setOrigin()`

Definition at line 345 of file `qwt_dial.cpp`.

6.18.4.8 `QwtDial::Mode` `QwtDial::mode () const`

Returns:

mode of the dial.

The value of the dial is indicated by the difference between the origin and the direction of the needle. In case of `QwtDial::RotateNeedle` the scale arc is fixed to the `origin()` and the needle is rotating, in case of `QwtDial::RotateScale`, the needle points to `origin()` and the scale is rotating.

The default mode is `QwtDial::RotateNeedle`.

See also:

`setMode()`, `origin()`, `setScaleArc()`, `value()`

Definition at line 368 of file `qwt_dial.cpp`.

Referenced by `drawContents()`, `QwtCompass::drawScaleContents()`, `getValue()`, and `QwtCompass::keyPressEvent()`.

6.18.4.9 `void QwtDial::setWrapping (bool wrapping) [virtual]`

Sets whether it is possible to step the value from the highest value to the lowest value and vice versa to on.

Parameters:

wrapping en/disables wrapping

See also:

`wrapping()`, `QwtDoubleRange::periodic()`

Note:

The meaning of wrapping is like the wrapping property of `QSpinBox`, but not like it is used in `QDial`.

Definition at line 383 of file `qwt_dial.cpp`.

References `QwtDoubleRange::setPeriodic()`.

6.18.4.10 bool QwtDial::wrapping () const

[wrapping\(\)](#) holds whether it is possible to step the value from the highest value to the lowest value and vice versa.

See also:

[setWrapping\(\)](#), [QwtDoubleRange::setPeriodic\(\)](#)

Note:

The meaning of wrapping is like the wrapping property of QSpinBox, but not like it is used in QDial.

Definition at line 396 of file qwt_dial.cpp.

References [QwtDoubleRange::periodic\(\)](#).

Referenced by [getValue\(\)](#).

6.18.4.11 void QwtDial::setScale (int *maxMajIntv*, int *maxMinIntv*, double *step* = 0.0) [virtual]

Change the intervals of the scale

See also:

[QwtAbstractScaleDraw::setScale](#)

Definition at line 846 of file qwt_dial.cpp.

References [updateScale\(\)](#).

6.18.4.12 void QwtDial::setScaleArc (double *minArc*, double *maxArc*)

Change the arc of the scale

Parameters:

- minArc* Lower limit
- maxArc* Upper limit

Definition at line 961 of file qwt_dial.cpp.

6.18.4.13 void QwtDial::setScaleOptions (int *options*)

A wrapper method for accessing the scale draw.

- `options == 0`
No visible scale: [setScaleDraw\(NULL\)](#)
- `options & ScaleBackbone`
En/disable the backbone of the scale.
- `options & ScaleTicks`
En/disable the ticks of the scale.

- options & ScaleLabel
En/disable scale labels

See also:

[QwtAbstractScaleDraw::enableComponent](#)

Definition at line 869 of file qwt_dial.cpp.

References [QwtAbstractScaleDraw::enableComponent\(\)](#), and [setScaleDraw\(\)](#).

6.18.4.14 void QwtDial::setScaleTicks (int *minLen*, int *medLen*, int *majLen*, int *penWidth* = 1)

See: [QwtAbstractScaleDraw::setTickLength](#), [QwtDialScaleDraw::setPenWidth](#).

Definition at line 889 of file qwt_dial.cpp.

References [QwtDialScaleDraw::setPenWidth\(\)](#), and [QwtAbstractScaleDraw::setTickLength\(\)](#).

6.18.4.15 double QwtDial::minScaleArc () const

Returns:

Lower limit of the scale arc

Definition at line 919 of file qwt_dial.cpp.

6.18.4.16 double QwtDial::maxScaleArc () const

Returns:

Upper limit of the scale arc

Definition at line 925 of file qwt_dial.cpp.

6.18.4.17 void QwtDial::setOrigin (double *origin*) [virtual]

Change the origin.

The origin is the angle where scale and needle is relative to.

Parameters:

origin New origin

See also:

[origin\(\)](#)

Definition at line 938 of file qwt_dial.cpp.

6.18.4.18 double QwtDial::origin () const

The origin is the angle where scale and needle is relative to.

Returns:

Origin of the dial

See also:

[setOrigin\(\)](#)

Definition at line 950 of file qwt_dial.cpp.

Referenced by drawContents(), QwtAnalogClock::drawNeedle(), QwtCompass::drawScaleContents(), and QwtCompass::keyPressEvent().

6.18.4.19 void QwtDial::setNeedle (QwtDialNeedle * *needle*) [virtual]

Set a needle for the dial

Qwt is missing a set of good looking needles. Contributions are very welcome.

Parameters:

needle Needle

Warning:

The needle will be deleted, when a different needle is set or in `~QwtDial()`

Definition at line 756 of file qwt_dial.cpp.

References `needle()`.

6.18.4.20 const QwtDialNeedle * QwtDial::needle () const

Returns:

needle

See also:

[setNeedle\(\)](#)

Definition at line 772 of file qwt_dial.cpp.

Referenced by QwtAnalogClock::drawHand(), QwtAnalogClock::setHand(), and `setNeedle()`.

6.18.4.21 QwtDialNeedle * QwtDial::needle ()

Returns:

needle

See also:

[setNeedle\(\)](#)

Definition at line 781 of file qwt_dial.cpp.

6.18.4.22 QRect QwtDial::boundingRect () const

Returns:

bounding rect of the dial including the frame

See also:

[setLineWidth\(\)](#), [scaleContentsRect\(\)](#), [contentsRect\(\)](#)

Definition at line 297 of file qwt_dial.cpp.

Referenced by [contentsRect\(\)](#), [drawContents\(\)](#), [drawFrame\(\)](#), and [updateMask\(\)](#).

6.18.4.23 QRect QwtDial::contentsRect () const

Returns:

bounding rect of the circle inside the frame

See also:

[setLineWidth\(\)](#), [scaleContentsRect\(\)](#), [boundingRect\(\)](#)

Definition at line 280 of file qwt_dial.cpp.

References [boundingRect\(\)](#), and [lineWidth\(\)](#).

Referenced by [drawFocusIndicator\(\)](#), [getScrollMode\(\)](#), and [scaleContentsRect\(\)](#).

6.18.4.24 QRect QwtDial::scaleContentsRect () const [virtual]

Returns:

rect inside the scale

See also:

[setLineWidth\(\)](#), [boundingRect\(\)](#), [contentsRect\(\)](#)

Definition at line 310 of file qwt_dial.cpp.

References [contentsRect\(\)](#).

Referenced by [drawContents\(\)](#).

6.18.4.25 QSize QwtDial::sizeHint () const [virtual]

Returns:

Size hint

Definition at line 986 of file qwt_dial.cpp.

References [lineWidth\(\)](#).

6.18.4.26 `QSize QwtDial::minimumSizeHint () const` [virtual]

Return a minimum size hint.

Warning:

The return value of `QwtDial::minimumSizeHint()` depends on the font and the scale.

Definition at line 1002 of file `qwt_dial.cpp`.

References `lineWidth()`.

6.18.4.27 `void QwtDial::setScaleDraw (QwtDialScaleDraw * scaleDraw)` [virtual]

Set an individual scale draw

Parameters:

scaleDraw Scale draw

Warning:

The previous scale draw is deleted

Definition at line 829 of file `qwt_dial.cpp`.

References `scaleDraw()`, and `updateScale()`.

Referenced by `setScaleOptions()`.

6.18.4.28 `QwtDialScaleDraw * QwtDial::scaleDraw ()`

Return the scale draw.

Definition at line 812 of file `qwt_dial.cpp`.

Referenced by `setScaleDraw()`.

6.18.4.29 `const QwtDialScaleDraw * QwtDial::scaleDraw () const`

Return the scale draw.

Definition at line 818 of file `qwt_dial.cpp`.

6.18.4.30 `void QwtDial::paintEvent (QPaintEvent * e)` [protected, virtual]

Paint the dial

Parameters:

e Paint event

Definition at line 417 of file `qwt_dial.cpp`.

References `drawContents()`, `drawFocusIndicator()`, and `drawFrame()`.

6.18.4.31 void QwtDial::resizeEvent (QResizeEvent * *e*) [protected, virtual]

Resize the dial widget

Parameters:

e Resize event

Definition at line 405 of file qwt_dial.cpp.

References [hasVisibleBackground\(\)](#), and [updateMask\(\)](#).

6.18.4.32 void QwtDial::keyPressEvent (QKeyEvent * *e*) [protected, virtual]

Handles key events

- [Key_Down](#), [KeyLeft](#)
Decrement by 1
- [Key_Prior](#)
Decrement by [pageSize\(\)](#)
- [Key_Home](#)
Set the value to [minValue\(\)](#)
- [Key_Up](#), [KeyRight](#)
Increment by 1
- [Key_Next](#)
Increment by [pageSize\(\)](#)
- [Key_End](#)
Set the value to [maxValue\(\)](#)

See also:

[isReadOnly\(\)](#)

Reimplemented from [QwtAbstractSlider](#).

Reimplemented in [QwtCompass](#).

Definition at line 1172 of file qwt_dial.cpp.

References [QwtDoubleRange::incValue\(\)](#), [QwtAbstractSlider::isReadOnly\(\)](#), [QwtAbstractSlider::isValid\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), [QwtDoubleRange::pageSize\(\)](#), [QwtDoubleRange::prevValue\(\)](#), [QwtAbstractSlider::setValue\(\)](#), [QwtAbstractSlider::sliderMoved\(\)](#), and [QwtDoubleRange::value\(\)](#).

Referenced by [QwtCompass::keyPressEvent\(\)](#).

6.18.4.33 void QwtDial::updateMask () [protected, virtual]

Update the mask of the dial.

In case of "[hasVisibleBackground\(\)](#) == false", the background is transparent by a mask.

See also:

[showBackground\(\)](#), [hasVisibleBackground\(\)](#)

Definition at line 1231 of file qwt_dial.cpp.

References [boundingRect\(\)](#).

Referenced by [resizeEvent\(\)](#), and [showBackground\(\)](#).

6.18.4.34 void QwtDial::drawFrame (QPainter * *painter*) [protected, virtual]

Draw the frame around the dial

Parameters:

painter Painter

See also:

[lineWidth\(\)](#), [frameShadow\(\)](#)

Definition at line 495 of file qwt_dial.cpp.

References [boundingRect\(\)](#), [QwtPainter::drawRoundFrame\(\)](#), and [lineWidth\(\)](#).

Referenced by [paintEvent\(\)](#).

6.18.4.35 void QwtDial::drawContents (QPainter * *painter*) const [protected, virtual]

Draw the contents inside the frame.

[QColorGroup::Background](#) is the background color outside of the frame. [QColorGroup::Base](#) is the background color inside the frame. [QColorGroup::Foreground](#) is the background color inside the scale.

Parameters:

painter Painter

See also:

[boundingRect\(\)](#), [contentsRect\(\)](#), [scaleContentsRect\(\)](#), [QWidget::setPalette](#)

Definition at line 557 of file qwt_dial.cpp.

References [boundingRect\(\)](#), [drawNeedle\(\)](#), [drawScale\(\)](#), [drawScaleContents\(\)](#), [QwtAbstractSlider::isValid\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), [mode\(\)](#), [origin\(\)](#), [scaleContentsRect\(\)](#), and [QwtDoubleRange::value\(\)](#).

Referenced by [paintEvent\(\)](#).

6.18.4.36 void QwtDial::drawFocusIndicator (QPainter * *painter*) const [protected, virtual]

Draw a dotted round circle, if [!isReadOnly\(\)](#)

Parameters:

painter Painter

Definition at line 448 of file qwt_dial.cpp.

References `contentsRect()`, and `QwtAbstractSlider::isReadOnly()`.

Referenced by `paintEvent()`.

6.18.4.37 `void QwtDial::drawScale (QPainter * painter, const QPoint & center, int radius, double origin, double minArc, double maxArc) const` `[protected, virtual]`

Draw the scale

Parameters:

painter Painter

center Center of the dial

radius Radius of the scale

origin Origin of the scale

minArc Minimum of the arc

maxArc Minimum of the arc

See also:

`QwtAbstractScaleDraw::setAngleRange`

Definition at line 689 of file qwt_dial.cpp.

Referenced by `drawContents()`.

6.18.4.38 `void QwtDial::drawScaleContents (QPainter * painter, const QPoint & center, int radius) const` `[protected, virtual]`

Draw the contents inside the scale

Paints nothing.

Parameters:

painter Painter

center Center of the contents circle

radius Radius of the contents circle

Reimplemented in [QwtCompass](#).

Definition at line 740 of file qwt_dial.cpp.

Referenced by `drawContents()`.

6.18.4.39 `void QwtDial::drawNeedle (QPainter * painter, const QPoint & center, int radius, double direction, QPalette::ColorGroup cg) const` `[protected, virtual]`

Draw the needle

Parameters:

painter Painter

center Center of the dial

radius Length for the needle

direction Direction of the needle in degrees, counter clockwise

cg ColorGroup

Reimplemented in [QwtAnalogClock](#).

Definition at line 667 of file qwt_dial.cpp.

Referenced by drawContents().

6.18.4.40 [QwtText](#) QwtDial::scaleLabel (double *value*) const [protected, virtual]

Find the label for a value

Parameters:

value Value

Returns:

label

Reimplemented in [QwtAnalogClock](#), and [QwtCompass](#).

Definition at line 908 of file qwt_dial.cpp.

Referenced by QwtDialScaleDraw::label().

6.18.4.41 void QwtDial::updateScale () [protected]

Update the scale with the current attributes

See also:

[setScale\(\)](#)

Definition at line 796 of file qwt_dial.cpp.

References [QwtLinearScaleEngine::divideScale\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), and [QwtLinearScaleEngine::transformation\(\)](#).

Referenced by rangeChange(), setScale(), and setScaleDraw().

6.18.4.42 void QwtDial::rangeChange () [protected, virtual]

[QwtDoubleRange](#) update hook.

Reimplemented from [QwtDoubleRange](#).

Definition at line 787 of file qwt_dial.cpp.

References updateScale().

6.18.4.43 void QwtDial::valueChange () [protected, virtual]

[QwtDoubleRange](#) update hook.

Reimplemented from [QwtAbstractSlider](#).

Definition at line 977 of file qwt_dial.cpp.

References [QwtAbstractSlider::valueChange\(\)](#).

6.18.4.44 double QwtDial::getValue (const QPoint & pos) [protected, virtual]

Find the value for a given position

Parameters:

pos Position

Returns:

Value

Implements [QwtAbstractSlider](#).

Definition at line 1037 of file qwt_dial.cpp.

References [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), [mode\(\)](#), [QwtAbstractSlider::mouseOffset\(\)](#), [QwtAbstractSlider::scrollMode\(\)](#), [QwtAbstractSlider::setMouseOffset\(\)](#), [QwtDoubleRange::value\(\)](#), and [wrapping\(\)](#).

6.18.4.45 void QwtDial::getScrollMode (const QPoint & p, int & scrollMode, int & direction) [protected, virtual]

See also:

[QwtAbstractSlider::getScrollMode](#)

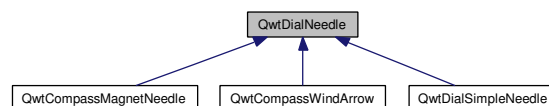
Implements [QwtAbstractSlider](#).

Definition at line 1140 of file qwt_dial.cpp.

References [contentsRect\(\)](#).

6.19 QwtDialNeedle Class Reference

Inheritance diagram for QwtDialNeedle:



6.19.1 Detailed Description

Base class for needles that can be used in a [QwtDial](#).

[QwtDialNeedle](#) is a pointer that indicates a value by pointing to a specific direction.

Qwt is missing a set of good looking needles. Contributions are very welcome.

See also:

[QwtDial](#), [QwtCompass](#)

Definition at line 31 of file qwt_dial_needle.h.

Public Member Functions

- [QwtDialNeedle](#) ()
- virtual [~QwtDialNeedle](#) ()
- virtual void [draw](#) (QPainter *painter, const QPoint ¢er, int length, double direction, QPalette::ColorGroup cg=QPalette::Active) const=0
- virtual void [setPalette](#) (const QPalette &)
- const QPalette & [palette](#) () const

Static Protected Member Functions

- static void [drawKnob](#) (QPainter *, const QPoint &pos, int width, const QBrush &, bool sunken)

6.19.2 Constructor & Destructor Documentation

6.19.2.1 QwtDialNeedle::QwtDialNeedle ()

Constructor.

Definition at line 25 of file qwt_dial_needle.cpp.

6.19.2.2 QwtDialNeedle::~~QwtDialNeedle () [virtual]

Destructor.

Definition at line 31 of file qwt_dial_needle.cpp.

6.19.3 Member Function Documentation

6.19.3.1 virtual void QwtDialNeedle::draw (QPainter * *painter*, const QPoint & *center*, int *length*, double *direction*, QPalette::ColorGroup *cg* = QPalette::Active) const [pure virtual]

Draw the needle

Parameters:

painter Painter

center Center of the dial, start position for the needle

length Length of the needle

direction Direction of the needle, in degrees counter clockwise

cg Color group, used for painting

Implemented in [QwtDialSimpleNeedle](#), [QwtCompassMagnetNeedle](#), and [QwtCompassWindArrow](#).

Referenced by [QwtAnalogClock::drawHand\(\)](#).

6.19.3.2 void QwtDialNeedle::setPalette (const QPalette & *palette*) [virtual]

Sets the palette for the needle.

Parameters:

palette New Palette

Definition at line 40 of file qwt_dial_needle.cpp.

Referenced by QwtCompassMagnetNeedle::QwtCompassMagnetNeedle(), QwtCompassWindArrow::QwtCompassWindArrow(), and QwtDialSimpleNeedle::QwtDialSimpleNeedle().

6.19.3.3 const QPalette & QwtDialNeedle::palette () const**Returns:**

the palette of the needle.

Definition at line 48 of file qwt_dial_needle.cpp.

Referenced by QwtCompassWindArrow::draw(), QwtCompassMagnetNeedle::draw(), QwtDialSimpleNeedle::draw(), QwtCompassMagnetNeedle::QwtCompassMagnetNeedle(), QwtCompassWindArrow::QwtCompassWindArrow(), and QwtDialSimpleNeedle::QwtDialSimpleNeedle().

6.19.3.4 void QwtDialNeedle::drawKnob (QPainter *, const QPoint & *pos*, int *width*, const QBrush &, bool *sunken*) [static, protected]

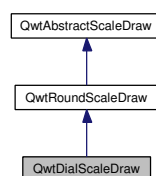
Draw the knob.

Definition at line 54 of file qwt_dial_needle.cpp.

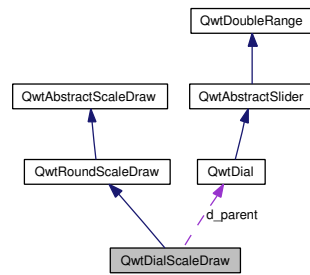
Referenced by QwtDialSimpleNeedle::drawArrowNeedle(), QwtDialSimpleNeedle::drawRayNeedle(), and QwtCompassMagnetNeedle::drawThinNeedle().

6.20 QwtDialScaleDraw Class Reference

Inheritance diagram for QwtDialScaleDraw:



Collaboration diagram for QwtDialScaleDraw:



6.20.1 Detailed Description

A special scale draw made for [QwtDial](#).

See also:

[QwtDial](#), [QwtCompass](#)

Definition at line 29 of file `qwt_dial.h`.

Public Member Functions

- [QwtDialScaleDraw](#) ([QwtDial](#) *)
- virtual [QwtText](#) label (double value) const
- void setPenWidth (uint)
- uint penWidth () const

6.20.2 Constructor & Destructor Documentation

6.20.2.1 [QwtDialScaleDraw::QwtDialScaleDraw](#) ([QwtDial](#) *parent) [explicit]

Constructor

Parameters:

parent Parent dial widget

Definition at line 77 of file `qwt_dial.cpp`.

6.20.3 Member Function Documentation

6.20.3.1 [QwtText](#) [QwtDialScaleDraw::label](#) (double value) const [virtual]

Call [QwtDial::scaleLabel](#) of the parent dial widget.

Parameters:

value Value to display

See also:

[QwtDial::scaleLabel](#)

Reimplemented from [QwtAbstractScaleDraw](#).

Definition at line 111 of file qwt_dial.cpp.

References [QwtAbstractScaleDraw::label\(\)](#), and [QwtDial::scaleLabel\(\)](#).

6.20.3.2 void QwtDialScaleDraw::setPenWidth (uint *penWidth*)

Set the pen width used for painting the scale

Parameters:

penWidth Pen width

See also:

[penWidth\(\)](#), [QwtDial::drawScale\(\)](#)

Definition at line 90 of file qwt_dial.cpp.

Referenced by [QwtDial::setScaleTicks\(\)](#).

6.20.3.3 uint QwtDialScaleDraw::penWidth () const

Returns:

Pen width used for painting the scale

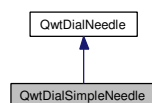
See also:

[setPenWidth](#), [QwtDial::drawScale\(\)](#)

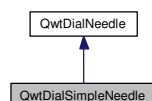
Definition at line 99 of file qwt_dial.cpp.

6.21 QwtDialSimpleNeedle Class Reference

Inheritance diagram for QwtDialSimpleNeedle:



Collaboration diagram for QwtDialSimpleNeedle:



6.21.1 Detailed Description

A needle for dial widgets.

The following colors are used:

- QColorGroup::Mid
Pointer
- QColorGroup::base
Knob

See also:

[QwtDial](#), [QwtCompass](#)

Definition at line 73 of file qwt_dial_needle.h.

Public Types

- enum [Style](#) {
 Arrow,
 Ray,
 TriangleStyle,
 ThinStyle,
 Style1,
 Style2,
 NoSymbol = -1,
 Ellipse,
 Rect,
 Diamond,
 Triangle,
 DTriangle,
 UTriangle,
 LTriangle,
 RTriangle,
 Cross,
 XCross,
 HLine,
 VLine,
 Star1,
 Star2,
 Hexagon,
 StyleCnt }

Public Member Functions

- [QwtDialSimpleNeedle](#) ([Style](#), bool hasKnob=true, const QColor &mid=Qt::gray, const QColor &base=Qt::darkGray)
- virtual void [draw](#) (QPainter *, const QPoint &, int length, double direction, QPalette::ColorGroup=QPalette::Active) const
- void [setWidth](#) (int width)
- int [width](#) () const

Static Public Member Functions

- static void [drawArrowNeedle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, int width, double direction, bool hasKnob)
- static void [drawRayNeedle](#) (QPainter *, const QPalette &, QPalette::ColorGroup, const QPoint &, int length, int width, double direction, bool hasKnob)

6.21.2 Member Enumeration Documentation

6.21.2.1 enum [QwtDialSimpleNeedle::Style](#)

Style of the needle.

Definition at line 77 of file qwt_dial_needle.h.

6.21.3 Constructor & Destructor Documentation

6.21.3.1 [QwtDialSimpleNeedle::QwtDialSimpleNeedle](#) ([Style](#) *style*, bool *hasKnob* = true, const QColor & *mid* = Qt::gray, const QColor & *base* = Qt::darkGray)

Constructor

Definition at line 91 of file qwt_dial_needle.cpp.

References [QwtDialNeedle::palette\(\)](#), and [QwtDialNeedle::setPalette\(\)](#).

6.21.4 Member Function Documentation

6.21.4.1 void [QwtDialSimpleNeedle::draw](#) (QPainter * *painter*, const QPoint & *center*, int *length*, double *direction*, QPalette::ColorGroup *colorGroup* = QPalette::Active) const [virtual]

Draw the needle

Parameters:

painter Painter

center Center of the dial, start position for the needle

length Length of the needle

direction Direction of the needle, in degrees counter clockwise

colorGroup Color group, used for painting

Implements [QwtDialNeedle](#).

Definition at line 132 of file qwt_dial_needle.cpp.

References [drawArrowNeedle\(\)](#), [drawRayNeedle\(\)](#), and [QwtDialNeedle::palette\(\)](#).

6.21.4.2 void QwtDialSimpleNeedle::drawArrowNeedle (QPainter * *painter*, const QPalette & *palette*, QPalette::ColorGroup *colorGroup*, const QPoint & *center*, int *length*, int *width*, double *direction*, bool *hasKnob*) [static]

Draw a needle looking like an arrow

Definition at line 202 of file qwt_dial_needle.cpp.

References QwtDialNeedle::drawKnob().

Referenced by draw().

6.21.4.3 void QwtDialSimpleNeedle::drawRayNeedle (QPainter * *painter*, const QPalette & *palette*, QPalette::ColorGroup *colorGroup*, const QPoint & *center*, int *length*, int *width*, double *direction*, bool *hasKnob*) [static]

Draw a needle looking like a ray

Definition at line 150 of file qwt_dial_needle.cpp.

References QwtDialNeedle::drawKnob().

Referenced by draw().

6.21.4.4 void QwtDialSimpleNeedle::setWidth (int *width*)

Set the width of the needle.

Definition at line 110 of file qwt_dial_needle.cpp.

6.21.4.5 int QwtDialSimpleNeedle::width () const

Returns:

the width of the needle

Definition at line 118 of file qwt_dial_needle.cpp.

6.22 QwtDoubleInterval Class Reference

6.22.1 Detailed Description

A class representing an interval.

The interval is represented by 2 doubles, the lower and the upper limit.

Definition at line 21 of file qwt_double_interval.h.

Public Member Functions

- [QwtDoubleInterval](#) ()
- [QwtDoubleInterval](#) (double min**Value**, double max**Value**)
- void [setInterval](#) (double min**Value**, double max**Value**)
- [QwtDoubleInterval normalized](#) () const
- [QwtDoubleInterval inverted](#) () const
- [QwtDoubleInterval limited](#) (double min**Value**, double max**Value**) const

- `int operator== (const QwtDoubleInterval &) const`
- `int operator!= (const QwtDoubleInterval &) const`
- `double minValue () const`
- `double maxValue () const`
- `double width () const`
- `void setMinValue (double)`
- `void setMaxValue (double)`
- `bool contains (double value) const`
- `bool intersects (const QwtDoubleInterval &) const`
- `QwtDoubleInterval intersect (const QwtDoubleInterval &) const`
- `QwtDoubleInterval unite (const QwtDoubleInterval &) const`
- `QwtDoubleInterval operator| (const QwtDoubleInterval &) const`
- `QwtDoubleInterval operator & (const QwtDoubleInterval &) const`
- `QwtDoubleInterval & operator|= (const QwtDoubleInterval &)`
- `QwtDoubleInterval & operator &= (const QwtDoubleInterval &)`
- `QwtDoubleInterval extend (double value) const`
- `QwtDoubleInterval operator| (double) const`
- `QwtDoubleInterval & operator|= (double)`
- `bool isValid () const`
- `bool isNull () const`
- `void invalidate ()`
- `QwtDoubleInterval symmetrize (double value) const`

6.22.2 Constructor & Destructor Documentation

6.22.2.1 QwtDoubleInterval::QwtDoubleInterval () [inline]

Default Constructor.

Creates an invalid interval [0.0, -1.0]

See also:

[setInterval](#), [isValid](#)

Definition at line 77 of file `qwt_double_interval.h`.

Referenced by `extend()`, `intersect()`, `inverted()`, `limited()`, `normalized()`, `symmetrize()`, and `unite()`.

6.22.2.2 QwtDoubleInterval::QwtDoubleInterval (double *minValue*, double *maxValue*) [inline]

Constructor

Parameters:

minValue Minimum value

maxValue Maximum value

Definition at line 89 of file `qwt_double_interval.h`.

6.22.3 Member Function Documentation

6.22.3.1 `void QwtDoubleInterval::setInterval (double minValue, double maxValue)` [inline]

Assign the limits of the interval

Parameters:

minValue Minimum value

maxValue Maximum value

Definition at line 101 of file `qwt_double_interval.h`.

Referenced by `QwtLog10ScaleEngine::autoScale()`.

6.22.3.2 `QwtDoubleInterval QwtDoubleInterval::normalized () const`

Normalize the limits of the interval.

If `maxValue() < minValue()` the limits will be inverted.

Returns:

Normalized interval

See also:

[isValid](#), [inverted](#)

Definition at line 28 of file `qwt_double_interval.cpp`.

References `isValid()`, and `QwtDoubleInterval()`.

Referenced by `QwtLinearScaleEngine::autoScale()`, and `QwtLog10ScaleEngine::divideScale()`.

6.22.3.3 `QwtDoubleInterval QwtDoubleInterval::inverted () const`

Invert the limits of the interval

Returns:

Inverted interval

See also:

[normalized](#)

Definition at line 43 of file `qwt_double_interval.cpp`.

References `QwtDoubleInterval()`.

6.22.3.4 `QwtDoubleInterval QwtDoubleInterval::limited (double lBound, double hBound) const`

Limit the interval

Parameters:

lBound Lower limit

hBound Upper limit

Returns:

Limited interval

Definition at line 159 of file qwt_double_interval.cpp.

References `isValid()`, `maxValue()`, `minValue()`, and `QwtDoubleInterval()`.

Referenced by `QwtLog10ScaleEngine::autoScale()`, and `QwtLog10ScaleEngine::divideScale()`.

6.22.3.5 `int QwtDoubleInterval::operator==(const QwtDoubleInterval &) const` `[inline]`

Compare two intervals.

Definition at line 172 of file qwt_double_interval.h.

References `d_maxValue`, and `d_minValue`.

6.22.3.6 `int QwtDoubleInterval::operator!=(const QwtDoubleInterval &) const` `[inline]`

Compare two intervals.

Definition at line 179 of file qwt_double_interval.h.

6.22.3.7 `double QwtDoubleInterval::minValue () const` `[inline]`

Returns:

Lower limit of the interval

Definition at line 128 of file qwt_double_interval.h.

Referenced by `QwtLinearScaleEngine::align()`, `QwtLog10ScaleEngine::autoScale()`, `QwtLinearScaleEngine::autoScale()`, `QwtLinearColorMap::colorIndex()`, `QwtColorMap::colorTable()`, `QwtScaleEngine::contains()`, `QwtLog10ScaleEngine::divideScale()`, `intersect()`, `intersects()`, `limited()`, `QwtLog10ScaleEngine::log10()`, `QwtLog10ScaleEngine::pow10()`, `QwtAlphaColorMap::rgb()`, `QwtLinearColorMap::rgb()`, `QwtScaleDiv::setInterval()`, `QwtAbstractScale::setScale()`, `unite()`, and `QwtPlot::updateAxes()`.

6.22.3.8 `double QwtDoubleInterval::maxValue () const` `[inline]`

Returns:

Upper limit of the interval

Definition at line 134 of file qwt_double_interval.h.

Referenced by `QwtLinearScaleEngine::align()`, `QwtLog10ScaleEngine::autoScale()`, `QwtLinearScaleEngine::autoScale()`, `QwtLinearColorMap::colorIndex()`, `QwtScaleEngine::contains()`, `QwtLog10ScaleEngine::divideScale()`, `intersect()`, `intersects()`, `limited()`, `QwtLog10ScaleEngine::log10()`, `QwtLog10ScaleEngine::pow10()`, `QwtScaleDiv::setInterval()`, `QwtAbstractScale::setScale()`, `unite()`, and `QwtPlot::updateAxes()`.

6.22.3.9 double QwtDoubleInterval::width () const [inline]

Return the width of an interval The width of invalid intervals is 0.0, otherwise the result is [maxValue\(\)](#) - [minValue\(\)](#).

See also:

[isValid](#)

Definition at line 146 of file `qwt_double_interval.h`.

References [isValid\(\)](#).

Referenced by [QwtLog10ScaleEngine::autoScale\(\)](#), [QwtLinearScaleEngine::autoScale\(\)](#), [QwtLinearColorMap::colorIndex\(\)](#), [QwtColorMap::colorTable\(\)](#), [QwtScaleEngine::contains\(\)](#), [QwtLog10ScaleEngine::divideScale\(\)](#), [QwtLinearScaleEngine::divideScale\(\)](#), [QwtAlphaColorMap::rgb\(\)](#), and [QwtLinearColorMap::rgb\(\)](#).

6.22.3.10 void QwtDoubleInterval::setMinValue (double *minValue*) [inline]

Assign the lower limit of the interval

Parameters:

minValue Minimum value

Definition at line 112 of file `qwt_double_interval.h`.

Referenced by [QwtLinearScaleEngine::autoScale\(\)](#).

6.22.3.11 void QwtDoubleInterval::setMaxValue (double *maxValue*) [inline]

Assign the upper limit of the interval

Parameters:

maxValue Maximum value

Definition at line 122 of file `qwt_double_interval.h`.

Referenced by [QwtLinearScaleEngine::autoScale\(\)](#).

6.22.3.12 bool QwtDoubleInterval::contains (double *value*) const

Test if a value is inside an interval

Parameters:

value Value

Returns:

true, if `value >= minValue\(\) && value <= maxValue\(\)`

Definition at line 54 of file `qwt_double_interval.cpp`.

References [isValid\(\)](#).

Referenced by [QwtRasterData::contourLines\(\)](#).

6.22.3.13 `bool QwtDoubleInterval::intersects (const QwtDoubleInterval & interval) const`

Test if two intervals overlap

Definition at line 119 of file qwt_double_interval.cpp.

References `isValid()`, `maxValue()`, and `minValue()`.

6.22.3.14 `QwtDoubleInterval QwtDoubleInterval::intersect (const QwtDoubleInterval &) const`

Intersect 2 intervals.

Definition at line 83 of file qwt_double_interval.cpp.

References `isValid()`, `maxValue()`, `minValue()`, and `QwtDoubleInterval()`.

Referenced by operator `&()`.

6.22.3.15 `QwtDoubleInterval QwtDoubleInterval::unite (const QwtDoubleInterval &) const`

Unite 2 intervals.

Definition at line 63 of file qwt_double_interval.cpp.

References `isValid()`, `maxValue()`, `minValue()`, and `QwtDoubleInterval()`.

Referenced by operator `|()`.

6.22.3.16 `QwtDoubleInterval QwtDoubleInterval::operator| (const QwtDoubleInterval & interval) const [inline]`

Union of two intervals

See also:

[unite](#)

Definition at line 165 of file qwt_double_interval.h.

References `unite()`.

6.22.3.17 `QwtDoubleInterval QwtDoubleInterval::operator & (const QwtDoubleInterval & interval) const [inline]`

Intersection of two intervals

See also:

[intersect](#)

Definition at line 155 of file qwt_double_interval.h.

References `intersect()`.

6.22.3.18 `QwtDoubleInterval QwtDoubleInterval::extend (double value) const`

Extend the interval

If value is below `minValue`, value becomes the lower limit. If value is above `maxValue`, value becomes the upper limit.

`extend` has no effect for invalid intervals

Parameters:

value Value

See also:

[isValid](#)

Definition at line 185 of file qwt_double_interval.cpp.

References [isValid\(\)](#), and [QwtDoubleInterval\(\)](#).

Referenced by [QwtLog10ScaleEngine::autoScale\(\)](#), [QwtLinearScaleEngine::autoScale\(\)](#), and [operator|\(\)](#).

6.22.3.19 [QwtDoubleInterval](#) QwtDoubleInterval::operator| (double *value*) const [inline]

Extend an interval

See also:

[extend](#)

Definition at line 188 of file qwt_double_interval.h.

References [extend\(\)](#).

6.22.3.20 bool QwtDoubleInterval::isValid () const [inline]**Returns:**

true, if [minValue\(\)](#) <= [maxValue\(\)](#)

Definition at line 200 of file qwt_double_interval.h.

Referenced by [QwtColorMap::colorTable\(\)](#), [QwtScaleEngine::contains\(\)](#), [contains\(\)](#), [QwtRasterData::contourLines\(\)](#), [extend\(\)](#), [intersect\(\)](#), [intersects\(\)](#), [limited\(\)](#), [normalized\(\)](#), [QwtScaleEngine::strip\(\)](#), [symmetrize\(\)](#), [unite\(\)](#), [QwtPlot::updateAxes\(\)](#), and [width\(\)](#).

6.22.3.21 bool QwtDoubleInterval::isNull () const [inline]**Returns:**

true, if [minValue\(\)](#) >= [maxValue\(\)](#)

Definition at line 194 of file qwt_double_interval.h.

6.22.3.22 void QwtDoubleInterval::invalidate () [inline]

Invalidate the interval

The limits are set to interval [0.0, -1.0]

See also:

[isValid](#)

Definition at line 211 of file qwt_double_interval.h.

6.22.3.23 QwtDoubleInterval QwtDoubleInterval::symmetrize (double *value*) const

Adjust the limit that is closer to *value*, so that *value* becomes the center of the interval.

Parameters:

value Center

Returns:

Interval with *value* as center

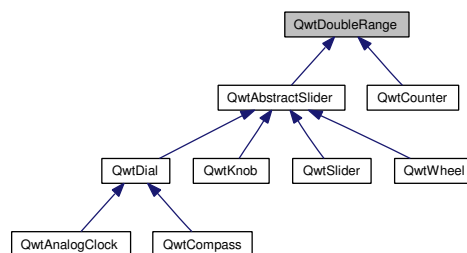
Definition at line 140 of file qwt_double_interval.cpp.

References `isValid()`, and `QwtDoubleInterval()`.

Referenced by `QwtLinearScaleEngine::autoScale()`.

6.23 QwtDoubleRange Class Reference

Inheritance diagram for QwtDoubleRange:



6.23.1 Detailed Description

A class which controls a value within an interval.

This class is useful as a base class or a member for sliders. It represents an interval of type double within which a value can be moved. The value can be either an arbitrary point inside the interval (see [QwtDoubleRange::setValue\(\)](#)), or it can be fitted into a step raster (see [QwtDoubleRange::fitValue\(\)](#) and [QwtDoubleRange::incValue\(\)](#)).

As a special case, a [QwtDoubleRange](#) can be periodic, which means that a value outside the interval will be mapped to a value inside the interval when [QwtDoubleRange::setValue\(\)](#), [QwtDoubleRange::fitValue\(\)](#), [QwtDoubleRange::incValue\(\)](#) or [QwtDoubleRange::incPages\(\)](#) are called.

Definition at line 31 of file qwt_double_range.h.

Public Member Functions

- [QwtDoubleRange](#) ()
- virtual [~QwtDoubleRange](#) ()
- void [setRange](#) (double vmin, double vmax, double vstep=0.0, int pagesize=1)
- void [setValid](#) (bool)
- bool [isValid](#) () const
- virtual void [setValue](#) (double)

- double [value](#) () const
- void [setPeriodic](#) (bool tf)
- bool [periodic](#) () const
- void [setStep](#) (double)
- double [step](#) () const
- double [maxValue](#) () const
- double [minValue](#) () const
- int [pageSize](#) () const
- virtual void [incValue](#) (int)
- virtual void [incPages](#) (int)
- virtual void [fitValue](#) (double)

Protected Member Functions

- double [exactValue](#) () const
- double [exactPrevValue](#) () const
- double [prevValue](#) () const
- virtual void [valueChange](#) ()
- virtual void [stepChange](#) ()
- virtual void [rangeChange](#) ()

6.23.2 Constructor & Destructor Documentation

6.23.2.1 QwtDoubleRange::QwtDoubleRange ()

The range is initialized to [0.0, 100.0], the step size to 1.0, and the value to 0.0.

Definition at line 22 of file `qwt_double_range.cpp`.

6.23.2.2 QwtDoubleRange::~QwtDoubleRange () [virtual]

Destroys the [QwtDoubleRange](#).

Definition at line 37 of file `qwt_double_range.cpp`.

6.23.3 Member Function Documentation

6.23.3.1 void QwtDoubleRange::setRange (double *vmin*, double *vmax*, double *vstep* = 0.0, int *pageSize* = 1)

Specify range and step size.

Parameters:

- vmin* lower boundary of the interval
- vmax* higher boundary of the interval
- vstep* step width
- pageSize* page size in steps

Warning:

- A change of the range changes the value if it lies outside the new range. The current value will *not* be adjusted to the new step raster.

- $v_{\max} < v_{\min}$ is allowed.
- If the step size is left out or set to zero, it will be set to 1/100 of the interval length.
- If the step size has an absurd value, it will be corrected to a better one.

Definition at line 172 of file `qwt_double_range.cpp`.

References `rangeChange()`, and `setStep()`.

Referenced by `QwtCounter::setMaxValue()`, and `QwtCounter::setMinValue()`.

6.23.3.2 void QwtDoubleRange::setValid (bool)

Set the value to be valid/invalid.

Reimplemented in [QwtAbstractSlider](#).

Definition at line 42 of file `qwt_double_range.cpp`.

References `valueChange()`.

Referenced by `QwtAbstractSlider::setValid()`.

6.23.3.3 bool QwtDoubleRange::isValid () const

Indicates if the value is valid.

Reimplemented in [QwtAbstractSlider](#).

Definition at line 52 of file `qwt_double_range.cpp`.

Referenced by `incPages()`, `incValue()`, and `QwtAbstractSlider::isValid()`.

6.23.3.4 void QwtDoubleRange::setValue (double x) [virtual]

Set a new value without adjusting to the step raster.

Parameters:

x new value

Warning:

The value is clipped when it lies outside the range. When the range is [QwtDoubleRange::periodic](#), it will be mapped to a point in the interval such that

```
new value := x + n * (max. value - min. value)
```

with an integer number *n*.

Reimplemented in [QwtAbstractSlider](#), and [QwtCounter](#).

Definition at line 150 of file `qwt_double_range.cpp`.

Referenced by `QwtCounter::setValue()`, and `QwtAbstractSlider::setValue()`.

6.23.3.5 double QwtDoubleRange::value () const

Returns the current value.

Reimplemented in [QwtCounter](#).

Definition at line 363 of file qwt_double_range.cpp.

Referenced by QwtDial::drawContents(), QwtAnalogClock::drawNeedle(), QwtCompass::drawScaleContents(), QwtSlider::drawSlider(), QwtWheel::drawWheel(), QwtSlider::getScrollMode(), QwtDial::getValue(), QwtDial::keyPressEvent(), QwtCompass::keyPressEvent(), QwtAbstractSlider::keyPressEvent(), QwtAbstractSlider::mouseMoveEvent(), QwtAbstractSlider::mousePressEvent(), QwtAbstractSlider::mouseReleaseEvent(), QwtAbstractSlider::timerEvent(), QwtCounter::value(), QwtAbstractSlider::valueChange(), and QwtAbstractSlider::wheelEvent().

6.23.3.6 void QwtDoubleRange::setPeriodic (bool *tf*)

Make the range periodic.

When the range is periodic, the value will be set to a point inside the interval such that

```
point = value + n * width
```

if the user tries to set a new value which is outside the range. If the range is nonperiodic (the default), values outside the range will be clipped.

Parameters:

tf true for a periodic range

Definition at line 251 of file qwt_double_range.cpp.

Referenced by QwtDial::setWrapping().

6.23.3.7 bool QwtDoubleRange::periodic () const

Returns true if the range is periodic.

See also:

[QwtDoubleRange::setPeriodic\(\)](#)

Definition at line 351 of file qwt_double_range.cpp.

Referenced by QwtDial::wrapping().

6.23.3.8 void QwtDoubleRange::setStep (double *vstep*)

Change the step raster.

Parameters:

vstep new step width

Warning:

The value will *not* be adjusted to the new step raster.

Reimplemented in [QwtCounter](#).

Definition at line 211 of file qwt_double_range.cpp.

References [stepChange\(\)](#).

Referenced by [setRange\(\)](#), and [QwtCounter::setStep\(\)](#).

6.23.3.9 double QwtDoubleRange::step () const

Returns:

the step size

See also:

[QwtDoubleRange::setStep](#), [QwtDoubleRange::setRange](#)

Reimplemented in [QwtCounter](#).

Definition at line 316 of file `qwt_double_range.cpp`.

Referenced by `QwtAbstractSlider::mouseReleaseEvent()`, `QwtCounter::step()`, and `QwtAbstractSlider::timerEvent()`.

6.23.3.10 double QwtDoubleRange::maxValue () const

Returns the value of the second border of the range.

`maxValue` returns the value which has been specified as the second parameter in [QwtDoubleRange::setRange](#).

See also:

[QwtDoubleRange::setRange\(\)](#)

Definition at line 329 of file `qwt_double_range.cpp`.

Referenced by `QwtDial::drawContents()`, `QwtWheel::drawWheel()`, `QwtWheel::getValue()`, `QwtDial::getValue()`, `QwtDial::keyPressEvent()`, `QwtCounter::keyPressEvent()`, `QwtCounter::maxVal()`, `QwtSlider::rangeChange()`, `QwtCounter::setMinValue()`, `QwtCounter::sizeHint()`, and `QwtDial::updateScale()`.

6.23.3.11 double QwtDoubleRange::minValue () const

Returns the value at the first border of the range.

`minValue` returns the value which has been specified as the first parameter in [setRange\(\)](#).

See also:

[QwtDoubleRange::setRange\(\)](#)

Definition at line 342 of file `qwt_double_range.cpp`.

Referenced by `QwtDial::drawContents()`, `QwtWheel::drawWheel()`, `QwtWheel::getValue()`, `QwtDial::getValue()`, `QwtDial::keyPressEvent()`, `QwtCounter::keyPressEvent()`, `QwtCounter::minVal()`, `QwtSlider::rangeChange()`, `QwtCounter::setMaxValue()`, `QwtCounter::sizeHint()`, and `QwtDial::updateScale()`.

6.23.3.12 int QwtDoubleRange::pageSize () const

Returns the page size in steps.

Definition at line 357 of file `qwt_double_range.cpp`.

Referenced by `QwtDial::keyPressEvent()`.

6.23.3.13 void QwtDoubleRange::incValue (int *nSteps*) [virtual]

Increment the value by a specified number of steps.

Parameters:

nSteps Number of steps to increment

Warning:

As a result of this operation, the new value will always be adjusted to the step raster.

Reimplemented in [QwtAbstractSlider](#).

Definition at line 262 of file qwt_double_range.cpp.

References [isValid\(\)](#).

Referenced by [QwtAbstractSlider::incValue\(\)](#), [QwtDial::keyPressEvent\(\)](#), [QwtCounter::keyPressEvent\(\)](#), [QwtAbstractSlider::keyPressEvent\(\)](#), and [QwtCounter::wheelEvent\(\)](#).

6.23.3.14 void QwtDoubleRange::incPages (int *nPages*) [virtual]

Increment the value by a specified number of pages.

Parameters:

nPages Number of pages to increment. A negative number decrements the value.

Warning:

The Page size is specified in the constructor.

Definition at line 274 of file qwt_double_range.cpp.

References [isValid\(\)](#).

Referenced by [QwtAbstractSlider::mouseReleaseEvent\(\)](#), [QwtAbstractSlider::timerEvent\(\)](#), and [QwtAbstractSlider::wheelEvent\(\)](#).

6.23.3.15 void QwtDoubleRange::fitValue (double *x*) [virtual]

Adjust the value to the closest point in the step raster.

Parameters:

x value

Warning:

The value is clipped when it lies outside the range. When the range is [QwtDoubleRange::periodic](#), it will be mapped to a point in the interval such that

```
new value := x + n * (max. value - min. value)
```

with an integer number *n*.

Reimplemented in [QwtAbstractSlider](#).

Definition at line 135 of file qwt_double_range.cpp.

Referenced by [QwtAbstractSlider::fitValue\(\)](#), [QwtAbstractSlider::mouseReleaseEvent\(\)](#), [QwtAbstractSlider::setPosition\(\)](#), and [QwtAbstractSlider::timerEvent\(\)](#).

6.23.3.16 double QwtDoubleRange::exactValue () const [protected]

Returns the exact value.

The exact value is the value which [QwtDoubleRange::value](#) would return if the value were not adjusted to the step raster. It differs from the current value only if [QwtDoubleRange::fitValue](#) or [QwtDoubleRange::incValue](#) have been used before. This function is intended for internal use in derived classes.

Definition at line 377 of file `qwt_double_range.cpp`.

Referenced by `QwtAbstractSlider::mouseMoveEvent()`, and `QwtAbstractSlider::timerEvent()`.

6.23.3.17 double QwtDoubleRange::exactPrevValue () const [protected]

Returns the exact previous value.

Definition at line 383 of file `qwt_double_range.cpp`.

Referenced by `QwtAbstractSlider::mouseMoveEvent()`.

6.23.3.18 double QwtDoubleRange::prevValue () const [protected]

Returns the previous value.

Definition at line 389 of file `qwt_double_range.cpp`.

Referenced by `QwtDial::keyPressEvent()`, `QwtAbstractSlider::keyPressEvent()`, `QwtAbstractSlider::mouseMoveEvent()`, and `QwtAbstractSlider::wheelEvent()`.

6.23.3.19 void QwtDoubleRange::valueChange () [protected, virtual]

Notify a change of value.

This virtual function is called whenever the value changes. The default implementation does nothing.

Reimplemented in [QwtAbstractSlider](#), [QwtDial](#), [QwtSlider](#), and [QwtWheel](#).

Definition at line 286 of file `qwt_double_range.cpp`.

Referenced by `setValid()`.

6.23.3.20 void QwtDoubleRange::stepChange () [protected, virtual]

Notify a change of the step size.

This virtual function is called whenever the step size changes. The default implementation does nothing.

Definition at line 308 of file `qwt_double_range.cpp`.

Referenced by `setStep()`.

6.23.3.21 void QwtDoubleRange::rangeChange () [protected, virtual]

Notify a change of the range.

This virtual function is called whenever the range changes. The default implementation does nothing.

Reimplemented in [QwtCounter](#), [QwtDial](#), and [QwtSlider](#).

Definition at line 297 of file `qwt_double_range.cpp`.

Referenced by `QwtSlider::rangeChange()`, and `setRange()`.

6.24 QwtDynGridLayout Class Reference

6.24.1 Detailed Description

The [QwtDynGridLayout](#) class lays out widgets in a grid, adjusting the number of columns and rows to the current size.

[QwtDynGridLayout](#) takes the space it gets, divides it up into rows and columns, and puts each of the widgets it manages into the correct cell(s). It lays out as many number of columns as possible (limited by [maxCols\(\)](#)).

Definition at line 32 of file `qwt_dyngrid_layout.h`.

Public Member Functions

- [QwtDynGridLayout](#) (QWidget *, int margin=0, int space=-1)
- [QwtDynGridLayout](#) (int space=-1)
- virtual [~QwtDynGridLayout](#) ()
- virtual void [invalidate](#) ()
- void [setMaxCols](#) (uint maxCols)
- uint [maxCols](#) () const
- uint [numRows](#) () const
- uint [numCols](#) () const
- virtual void [addItem](#) (QLayoutItem *)
- virtual QLayoutItem * [itemAt](#) (int index) const
- virtual QLayoutItem * [takeAt](#) (int index)
- virtual int [count](#) () const
- void [setExpandingDirections](#) (Qt::Orientations)
- virtual Qt::Orientations [expandingDirections](#) () const
- QList< QRect > [layoutItems](#) (const QRect &, uint numCols) const
- virtual int [maxItemWidth](#) () const
- virtual void [setGeometry](#) (const QRect &rect)
- virtual bool [hasHeightForWidth](#) () const
- virtual int [heightForWidth](#) (int) const
- virtual QSize [sizeHint](#) () const
- virtual bool [isEmpty](#) () const
- uint [itemCount](#) () const
- virtual uint [columnsForWidth](#) (int width) const

Protected Member Functions

- void [layoutGrid](#) (uint numCols, QwtArray< int > &rowHeight, QwtArray< int > &colWidth) const
- void [stretchGrid](#) (const QRect &rect, uint numCols, QwtArray< int > &rowHeight, QwtArray< int > &colWidth) const

6.24.2 Constructor & Destructor Documentation

6.24.2.1 [QwtDynGridLayout::QwtDynGridLayout](#) (QWidget * *parent*, int *margin* = 0, int *spacing* = -1) [`explicit`]

Parameters:

parent Parent widget

margin Margin

spacing Spacing

Definition at line 107 of file qwt_dyngrid_layout.cpp.

6.24.2.2 QwtDynGridLayout::QwtDynGridLayout (int *spacing* = -1) [explicit]**Parameters:**

spacing Spacing

Definition at line 133 of file qwt_dyngrid_layout.cpp.

6.24.2.3 QwtDynGridLayout::~QwtDynGridLayout () [virtual]

Destructor.

Definition at line 158 of file qwt_dyngrid_layout.cpp.

6.24.3 Member Function Documentation**6.24.3.1 void QwtDynGridLayout::setMaxCols (uint *maxCols*)**

Limit the number of columns.

Parameters:

maxCols upper limit, 0 means unlimited

See also:

[QwtDynGridLayout::maxCols\(\)](#)

Definition at line 194 of file qwt_dyngrid_layout.cpp.

6.24.3.2 uint QwtDynGridLayout::maxCols () const

Return the upper limit for the number of columns. 0 means unlimited, what is the default.

See also:

[QwtDynGridLayout::setMaxCols\(\)](#)

Definition at line 205 of file qwt_dyngrid_layout.cpp.

Referenced by columnsForWidth().

6.24.3.3 uint QwtDynGridLayout::numRows () const**Returns:**

Number of rows of the current layout.

See also:

[QwtDynGridLayout::numCols](#)

Warning:

The number of rows might change whenever the geometry changes

Definition at line 678 of file `qwt_dyngrid_layout.cpp`.

Referenced by `heightForWidth()`, `sizeHint()`, and `stretchGrid()`.

6.24.3.4 `uint QwtDynGridLayout::numCols () const`

Returns:

Number of columns of the current layout.

See also:

[QwtDynGridLayout::numRows](#)

Warning:

The number of columns might change whenever the geometry changes

Definition at line 688 of file `qwt_dyngrid_layout.cpp`.

Referenced by `columnsForWidth()`.

6.24.3.5 `void QwtDynGridLayout::addItem (QLayoutItem *) [virtual]`

Adds item to the next free position.

Definition at line 212 of file `qwt_dyngrid_layout.cpp`.

References `invalidate()`.

6.24.3.6 `QList< QRect > QwtDynGridLayout::layoutItems (const QRect & rect, uint numCols) const`

Calculate the geometries of the layout items for a layout with `numCols` columns and a given `rect`.

Parameters:

rect Rect where to place the items

numCols Number of columns

Returns:

item geometries

Definition at line 440 of file `qwt_dyngrid_layout.cpp`.

References `d_data`.

Referenced by `QwtPlot::printLegend()`, and `setGeometry()`.

6.24.3.7 `int QwtDynGridLayout::maxItemWidth () const` [virtual]**Returns:**

the maximum width of all layout items

Definition at line 409 of file `qwt_dyngrid_layout.cpp`.

References `isEmpty()`.

6.24.3.8 `void QwtDynGridLayout::setGeometry (const QRect & rect)` [virtual]

Reorganizes columns and rows and resizes managed widgets within the rectangle `rect`.

Definition at line 314 of file `qwt_dyngrid_layout.cpp`.

References `columnsForWidth()`, `isEmpty()`, `itemCount()`, and `layoutItems()`.

6.24.3.9 `bool QwtDynGridLayout::hasHeightForWidth () const` [virtual]**Returns:**

true: [QwtDynGridLayout](#) implements `heightForWidth`.

See also:

[QwtDynGridLayout::heightForWidth\(\)](#)

Definition at line 547 of file `qwt_dyngrid_layout.cpp`.

6.24.3.10 `int QwtDynGridLayout::heightForWidth (int width) const` [virtual]**Returns:**

The preferred height for this layout, given the width `w`.

See also:

[QwtDynGridLayout::hasHeightForWidth\(\)](#)

Definition at line 557 of file `qwt_dyngrid_layout.cpp`.

References `columnsForWidth()`, `isEmpty()`, `itemCount()`, `layoutGrid()`, and `numRows()`.

6.24.3.11 `QSize QwtDynGridLayout::sizeHint () const` [virtual]

Return the size hint. If [maxCols\(\)](#) > 0 it is the size for a grid with [maxCols\(\)](#) columns, otherwise it is the size for a grid with only one row.

See also:

[QwtDynGridLayout::maxCols\(\)](#), [QwtDynGridLayout::setMaxCols\(\)](#)

Definition at line 647 of file `qwt_dyngrid_layout.cpp`.

References `isEmpty()`, `itemCount()`, `layoutGrid()`, and `numRows()`.

6.24.3.12 bool QwtDynGridLayout::isEmpty () const [virtual]**Returns:**

true if this layout is empty.

Definition at line 222 of file qwt_dyngrid_layout.cpp.

Referenced by columnsForWidth(), heightForWidth(), maxItemWidth(), setGeometry(), sizeHint(), and stretchGrid().

6.24.3.13 uint QwtDynGridLayout::itemCount () const**Returns:**

number of layout items

Definition at line 231 of file qwt_dyngrid_layout.cpp.

Referenced by columnsForWidth(), heightForWidth(), setGeometry(), sizeHint(), and stretchGrid().

6.24.3.14 uint QwtDynGridLayout::columnsForWidth (int *width*) const [virtual]

Calculate the number of columns for a given width. It tries to use as many columns as possible (limited by [maxCols\(\)](#))

Parameters:

width Available width for all columns

See also:

[QwtDynGridLayout::maxCols\(\)](#), [QwtDynGridLayout::setMaxCols\(\)](#)

Definition at line 353 of file qwt_dyngrid_layout.cpp.

References isEmpty(), itemCount(), maxCols(), and numCols().

Referenced by heightForWidth(), QwtPlot::printLegend(), and setGeometry().

6.24.3.15 void QwtDynGridLayout::layoutGrid (uint *numCols*, QwtArray< int > & *rowHeight*, QwtArray< int > & *colWidth*) const [protected]

Calculate the dimensions for the columns and rows for a grid of numCols columns.

Parameters:

numCols Number of columns.

rowHeight Array where to fill in the calculated row heights.

colWidth Array where to fill in the calculated column widths.

Definition at line 518 of file qwt_dyngrid_layout.cpp.

Referenced by heightForWidth(), and sizeHint().

6.24.3.16 void QwtDynGridLayout::stretchGrid (const QRect & rect, uint numCols, QwtArray<int> & rowHeight, QwtArray<int> & colWidth) const [protected]

Stretch columns in case of expanding() & QSizePolicy::Horizontal and rows in case of expanding() & QSizePolicy::Vertical to fill the entire rect. Rows and columns are stretched with the same factor.

See also:

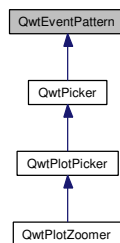
QwtDynGridLayout::setExpanding(), QwtDynGridLayout::expanding()

Definition at line 586 of file qwt_dyngrid_layout.cpp.

References expandingDirections(), isEmpty(), itemCount(), and numRows().

6.25 QwtEventPattern Class Reference

Inheritance diagram for QwtEventPattern:



6.25.1 Detailed Description

A collection of event patterns.

[QwtEventPattern](#) introduces an level of indirection for mouse and keyboard inputs. Those are represented by symbolic names, so the application code can be configured by individual mappings.

See also:

[QwtPicker](#), [QwtPickerMachine](#), [QwtPlotZoomer](#)

Definition at line 28 of file qwt_event_pattern.h.

Public Types

- enum [MousePatternCode](#) {
 - MouseSelect1,
 - MouseSelect2,
 - MouseSelect3,
 - MouseSelect4,
 - MouseSelect5,
 - MouseSelect6,
 - MousePatternCount }

- enum [KeyPatternCode](#) {
 KeySelect1,
 KeySelect2,
 KeyAbort,
 KeyLeft,
 KeyRight,
 KeyUp,
 KeyDown,
 KeyRedo,
 KeyUndo,
 KeyHome,
 KeyPatternCount }

Public Member Functions

- [QwtEventPattern](#) ()
- virtual [~QwtEventPattern](#) ()
- void [initMousePattern](#) (int numButtons)
- void [initKeyPattern](#) ()
- void [setMousePattern](#) (uint pattern, int button, int state=Qt::NoButton)
- void [setKeyPattern](#) (uint pattern, int key, int state=Qt::NoButton)
- void [setMousePattern](#) (const QwtArray< [MousePattern](#) > &)
- void [setKeyPattern](#) (const QwtArray< [KeyPattern](#) > &)
- const QwtArray< [MousePattern](#) > & [mousePattern](#) () const
- const QwtArray< [KeyPattern](#) > & [keyPattern](#) () const
- QwtArray< [MousePattern](#) > & [mousePattern](#) ()
- QwtArray< [KeyPattern](#) > & [keyPattern](#) ()
- bool [mouseMatch](#) (uint pattern, const QMouseEvent *) const
- bool [keyMatch](#) (uint pattern, const QKeyEvent *) const

Protected Member Functions

- virtual bool [mouseMatch](#) (const [MousePattern](#) &, const QMouseEvent *) const
- virtual bool [keyMatch](#) (const [KeyPattern](#) &, const QKeyEvent *) const

Classes

- class [KeyPattern](#)
 A pattern for key events.
- class [MousePattern](#)
 A pattern for mouse events.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum [QwtEventPattern::MousePatternCode](#)

Symbolic mouse input codes.

The default initialization for 3 button mice is:

- MouseSelect1
Qt::LeftButton
- MouseSelect2
Qt::RightButton
- MouseSelect3
Qt::MidButton
- MouseSelect4
Qt::LeftButton + Qt::ShiftButton
- MouseSelect5
Qt::RightButton + Qt::ShiftButton
- MouseSelect6
Qt::MidButton + Qt::ShiftButton

The default initialization for 2 button mice is:

- MouseSelect1
Qt::LeftButton
- MouseSelect2
Qt::RightButton
- MouseSelect3
Qt::LeftButton + Qt::AltButton
- MouseSelect4
Qt::LeftButton + Qt::ShiftButton
- MouseSelect5
Qt::RightButton + Qt::ShiftButton
- MouseSelect6
Qt::LeftButton + Qt::AltButton + Qt::ShiftButton

The default initialization for 1 button mice is:

- MouseSelect1
Qt::LeftButton
- MouseSelect2
Qt::LeftButton + Qt::ControlButton

- MouseSelect3
Qt::LeftButton + Qt::AltButton
- MouseSelect4
Qt::LeftButton + Qt::ShiftButton
- MouseSelect5
Qt::LeftButton + Qt::ControlButton + Qt::ShiftButton
- MouseSelect6
Qt::LeftButton + Qt::AltButton + Qt::ShiftButton

See also:

[initMousePattern\(\)](#)

Definition at line 79 of file qwt_event_pattern.h.

6.25.2.2 enum QwtEventPattern::KeyPatternCode

Symbolic keyboard input codes.

Default initialization:

- KeySelect1
Qt::Key_Return
- KeySelect2
Qt::Key_Space
- KeyAbort
Qt::Key_Escape
- KeyLeft
Qt::Key_Left
- KeyRight
Qt::Key_Right
- KeyUp
Qt::Key_Up
- KeyDown
Qt::Key_Down
- KeyUndo
Qt::Key_Minus
- KeyRedo
Qt::Key_Plus
- KeyHome
Qt::Key_Escape

Definition at line 118 of file qwt_event_pattern.h.

6.25.3 Constructor & Destructor Documentation

6.25.3.1 QwtEventPattern::QwtEventPattern ()

Constructor

See also:

[MousePatternCode](#), [KeyPatternCode](#)

Definition at line 19 of file qwt_event_pattern.cpp.

References `initKeyPattern()`, and `initMousePattern()`.

6.25.3.2 QwtEventPattern::~QwtEventPattern () [virtual]

Destructor.

Definition at line 28 of file qwt_event_pattern.cpp.

6.25.4 Member Function Documentation

6.25.4.1 void QwtEventPattern::initMousePattern (int *numButtons*)

Set default mouse patterns, depending on the number of mouse buttons

Parameters:

numButtons Number of mouse buttons (≤ 3)

See also:

[MousePatternCode](#)

Definition at line 38 of file qwt_event_pattern.cpp.

References `setMousePattern()`.

Referenced by `QwtEventPattern()`.

6.25.4.2 void QwtEventPattern::initKeyPattern ()

Set default mouse patterns.

See also:

[KeyPatternCode](#)

Definition at line 88 of file qwt_event_pattern.cpp.

References `setKeyPattern()`.

Referenced by `QwtEventPattern()`.

6.25.4.3 void QwtEventPattern::setMousePattern (uint *pattern*, int *button*, int *state* = Qt::NoButton)

Change one mouse pattern

Parameters:*pattern* Index of the pattern*button* Button*state* State**See also:**

QMouseEvent

Definition at line 115 of file qwt_event_pattern.cpp.

Referenced by initMousePattern().

6.25.4.4 void QwtEventPattern::setKeyPattern (uint *pattern*, int *key*, int *state* = Qt::NoButton)

Change one key pattern

Parameters:*pattern* Index of the pattern*key* Key*state* State**See also:**

QKeyEvent

Definition at line 133 of file qwt_event_pattern.cpp.

Referenced by initKeyPattern().

6.25.4.5 void QwtEventPattern::setMousePattern (const QwtArray< [MousePattern](#) > &)

Change the mouse event patterns.

Definition at line 143 of file qwt_event_pattern.cpp.

6.25.4.6 void QwtEventPattern::setKeyPattern (const QwtArray< [KeyPattern](#) > &)

Change the key event patterns.

Definition at line 149 of file qwt_event_pattern.cpp.

6.25.4.7 const QwtArray< [QwtEventPattern::MousePattern](#) > & QwtEventPattern::mousePattern () const

Return mouse patterns.

Definition at line 156 of file qwt_event_pattern.cpp.

6.25.4.8 const QwtArray< [QwtEventPattern::KeyPattern](#) > & QwtEventPattern::keyPattern () const

Return key patterns.

Definition at line 163 of file qwt_event_pattern.cpp.

6.25.4.9 `QwtArray< QwtEventPattern::MousePattern > & QwtEventPattern::mousePattern ()`

Return ,ouse patterns.

Definition at line 169 of file qwt_event_pattern.cpp.

6.25.4.10 `QwtArray< QwtEventPattern::KeyPattern > & QwtEventPattern::keyPattern ()`

Return Key patterns.

Definition at line 175 of file qwt_event_pattern.cpp.

6.25.4.11 `bool QwtEventPattern::mouseMatch (uint pattern, const QMouseEvent * e) const`

Compare a mouse event with an event pattern.

A mouse event matches the pattern when both have the same button value and in the state value the same key flags(Qt::KeyButtonMask) are set.

Parameters:

pattern Index of the event pattern

e Mouse event

Returns:

true if matches

See also:

[keyMatch\(\)](#)

Definition at line 193 of file qwt_event_pattern.cpp.

Referenced by QwtPickerPolygonMachine::transition(), QwtPickerDragRectMachine::transition(), QwtPickerClickRectMachine::transition(), QwtPickerDragPointMachine::transition(), QwtPickerClickPointMachine::transition(), and QwtPlotZoomer::widgetMouseReleaseEvent().

6.25.4.12 `bool QwtEventPattern::keyMatch (uint pattern, const QKeyEvent * e) const`

Compare a key event with an event pattern.

A key event matches the pattern when both have the same key value and in the state value the same key flags (Qt::KeyButtonMask) are set.

Parameters:

pattern Index of the event pattern

e Key event

Returns:

true if matches

See also:

[mouseMatch\(\)](#)

Definition at line 248 of file qwt_event_pattern.cpp.

Referenced by QwtPickerPolygonMachine::transition(), QwtPickerDragRectMachine::transition(), QwtPickerClickRectMachine::transition(), QwtPickerDragPointMachine::transition(), QwtPickerClickPointMachine::transition(), QwtPlotZoomer::widgetKeyPressEvent(), and QwtPicker::widgetKeyPressEvent().

6.25.4.13 `bool QwtEventPattern::mouseMatch (const MousePattern & pattern, const QMouseEvent * e) const` `[protected, virtual]`

Compare a mouse event with an event pattern.

A mouse event matches the pattern when both have the same button value and in the state value the same key flags (Qt::KeyButtonMask) are set.

Parameters:

pattern Mouse event pattern

e Mouse event

Returns:

true if matches

See also:

[keyMatch\(\)](#)

Definition at line 217 of file qwt_event_pattern.cpp.

References QwtEventPattern::MousePattern::button, and QwtEventPattern::MousePattern::state.

6.25.4.14 `bool QwtEventPattern::keyMatch (const KeyPattern & pattern, const QKeyEvent * e) const` `[protected, virtual]`

Compare a key event with an event pattern.

A key event matches the pattern when both have the same key value and in the state value the same key flags (Qt::KeyButtonMask) are set.

Parameters:

pattern Key event pattern

e Key event

Returns:

true if matches

See also:

[mouseMatch\(\)](#)

Definition at line 272 of file qwt_event_pattern.cpp.

References QwtEventPattern::KeyPattern::key, and QwtEventPattern::KeyPattern::state.

6.26 QwtEventPattern::KeyPattern Class Reference

6.26.1 Detailed Description

A pattern for key events.

Definition at line 151 of file qwt_event_pattern.h.

Public Member Functions

- [KeyPattern](#) (int k=0, int st=Qt::NoButton)

Public Attributes

- int [key](#)
- int [state](#)

6.27 QwtEventPattern::MousePattern Class Reference

6.27.1 Detailed Description

A pattern for mouse events.

Definition at line 137 of file qwt_event_pattern.h.

Public Member Functions

- [MousePattern](#) (int btn=Qt::NoButton, int st=Qt::NoButton)

Public Attributes

- int [button](#)
- int [state](#)

6.28 QwtIntervalData Class Reference

6.28.1 Detailed Description

Interval data class.

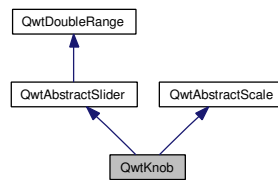
Definition at line 34 of file qwt_interval_data.h.

Public Member Functions

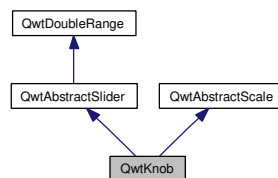
- [QwtIntervalData](#) ()
- [QwtIntervalData](#) (const QwtArray< [QwtDoubleInterval](#) > &, const QwtArray< double > &)
- void [setData](#) (const QwtArray< [QwtDoubleInterval](#) > &, const QwtArray< double > &)
- size_t [size](#) () const
- const [QwtDoubleInterval](#) & [interval](#) (size_t i) const
- double [value](#) (size_t i) const
- [QwtDoubleRect](#) [boundingRect](#) () const

6.29 QwtKnob Class Reference

Inheritance diagram for QwtKnob:



Collaboration diagram for QwtKnob:



6.29.1 Detailed Description

The Knob Widget.

The [QwtKnob](#) widget imitates look and behaviour of a volume knob on a radio. It contains a scale around the knob which is set up automatically or can be configured manually (see [QwtAbstractScale](#)). Automatic scrolling is enabled when the user presses a mouse button on the scale. For a description of signals, slots and other members, see [QwtAbstractSlider](#).

See also:

[QwtAbstractSlider](#) and [QwtAbstractScale](#) for the descriptions of the inherited members.

Definition at line 34 of file qwt_knob.h.

Public Types

- enum [Symbol](#) {
 Line,
 Dot }

Public Member Functions

- [QwtKnob](#) (QWidget *parent=NULL)
- virtual [~QwtKnob](#) ()
- void [setKnobWidth](#) (int w)
- int [knobWidth](#) () const
- void [setTotalAngle](#) (double angle)
- double [totalAngle](#) () const

- void [setBorderWidth](#) (int bw)
- int [borderWidth](#) () const
- void [setSymbol](#) (Symbol)
- Symbol [symbol](#) () const
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- void [setScaleDraw](#) (QwtRoundScaleDraw *)
- const QwtRoundScaleDraw * [scaleDraw](#) () const
- QwtRoundScaleDraw * [scaleDraw](#) ()

Protected Member Functions

- virtual void [paintEvent](#) (QPaintEvent *e)
- virtual void [resizeEvent](#) (QResizeEvent *e)
- void [draw](#) (QPainter *p, const QRect &r)
- void [drawKnob](#) (QPainter *p, const QRect &r)
- void [drawMarker](#) (QPainter *p, double arc, const QColor &c)

6.29.2 Member Enumeration Documentation

6.29.2.1 enum [QwtKnob::Symbol](#)

Symbol

See also:

[QwtKnob::QwtKnob\(\)](#)

Definition at line 49 of file qwt_knob.h.

6.29.3 Constructor & Destructor Documentation

6.29.3.1 [QwtKnob::QwtKnob \(QWidget *parent = NULL\)](#) [explicit]

Constructor

Parameters:

parent Parent widget

Definition at line 56 of file qwt_knob.cpp.

6.29.3.2 [QwtKnob::~~QwtKnob \(\)](#) [virtual]

Destructor.

Definition at line 96 of file qwt_knob.cpp.

6.29.4 Member Function Documentation

6.29.4.1 void QwtKnob::setKnobWidth (int *w*)

Change the knob's width.

The specified width must be ≥ 5 , or it will be clipped.

Parameters:

w New width

Definition at line 449 of file qwt_knob.cpp.

6.29.4.2 int QwtKnob::knobWidth () const

Return the width of the knob.

Definition at line 456 of file qwt_knob.cpp.

6.29.4.3 void QwtKnob::setTotalAngle (double *angle*)

Set the total angle by which the knob can be turned.

Parameters:

angle Angle in degrees.

The default angle is 270 degrees. It is possible to specify an angle of more than 360 degrees so that the knob can be turned several times around its axis.

Definition at line 131 of file qwt_knob.cpp.

References `scaleDraw()`, and `QwtRoundScaleDraw::setAngleRange()`.

Referenced by `setScaleDraw()`.

6.29.4.4 double QwtKnob::totalAngle () const

Return the total angle.

Definition at line 144 of file qwt_knob.cpp.

6.29.4.5 void QwtKnob::setBorderWidth (int *bw*)

Set the knob's border width.

Parameters:

bw new border width

Definition at line 465 of file qwt_knob.cpp.

6.29.4.6 int QwtKnob::borderWidth () const

Return the border width.

Definition at line 472 of file qwt_knob.cpp.

6.29.4.7 void QwtKnob::setSymbol (QwtKnob::Symbol s)

Set the symbol of the knob.

See also:

[symbol\(\)](#)

Definition at line 105 of file qwt_knob.cpp.

6.29.4.8 QwtKnob::Symbol QwtKnob::symbol () const

Returns:

symbol of the knob

See also:

[setSymbol\(\)](#)

Definition at line 118 of file qwt_knob.cpp.

6.29.4.9 QSize QwtKnob::sizeHint () const [virtual]

Returns:

[minimumSizeHint\(\)](#)

Definition at line 523 of file qwt_knob.cpp.

References [minimumSizeHint\(\)](#).

6.29.4.10 QSize QwtKnob::minimumSizeHint () const [virtual]

Return a minimum size hint.

Warning:

The return value of [QwtKnob::minimumSizeHint\(\)](#) depends on the font and the scale.

Definition at line 533 of file qwt_knob.cpp.

References [QwtRoundScaleDraw::extent\(\)](#), and [scaleDraw\(\)](#).

Referenced by [sizeHint\(\)](#).

6.29.4.11 void QwtKnob::setScaleDraw (QwtRoundScaleDraw * scaleDraw)

Change the scale draw of the knob

For changing the labels of the scales, it is necessary to derive from [QwtRoundScaleDraw](#) and overload [QwtRoundScaleDraw::label\(\)](#).

See also:

[scaleDraw\(\)](#)

Definition at line 158 of file qwt_knob.cpp.

References [scaleDraw\(\)](#), [QwtAbstractScale::setAbstractScaleDraw\(\)](#), and [setTotalAngle\(\)](#).

6.29.4.12 `const QwtRoundScaleDraw * QwtKnob::scaleDraw () const`**Returns:**

the scale draw of the knob

See also:

[setScaleDraw\(\)](#)

Definition at line 168 of file `qwt_knob.cpp`.

References `QwtAbstractScale::abstractScaleDraw()`.

Referenced by `draw()`, `minimumSizeHint()`, `setScaleDraw()`, and `setTotalAngle()`.

6.29.4.13 `QwtRoundScaleDraw * QwtKnob::scaleDraw ()`**Returns:**

the scale draw of the knob

See also:

[setScaleDraw\(\)](#)

Definition at line 177 of file `qwt_knob.cpp`.

References `QwtAbstractScale::abstractScaleDraw()`.

6.29.4.14 `void QwtKnob::paintEvent (QPaintEvent * e) [protected, virtual]`

Repaint the knob.

Definition at line 360 of file `qwt_knob.cpp`.

References `draw()`.

6.29.4.15 `void QwtKnob::resizeEvent (QResizeEvent * e) [protected, virtual]`

Qt Resize Event.

Definition at line 329 of file `qwt_knob.cpp`.

6.29.4.16 `void QwtKnob::draw (QPainter * p, const QRect & ur) [protected]`

Repaint the knob.

Definition at line 379 of file `qwt_knob.cpp`.

References `QwtAbstractScaleDraw::draw()`, `QwtPainter::drawFocusRect()`, `drawKnob()`, and `scaleDraw()`.

Referenced by `paintEvent()`.

6.29.4.17 `void QwtKnob::drawKnob (QPainter * painter, const QRect & r) [protected]`

Draw the knob.

Parameters:

- painter* painter
- r* Bounding rectangle of the knob (without scale)

Definition at line 187 of file qwt_knob.cpp.

References `drawMarker()`, and `QwtAbstractSlider::isValid()`.

Referenced by `draw()`.

6.29.4.18 void QwtKnob::drawMarker (QPainter * *p*, double *arc*, const QColor & *c*)
[protected]

Draw the marker at the knob's front.

Parameters:

- p* Painter
- arc* Angle of the marker
- c* Marker color

Definition at line 402 of file qwt_knob.cpp.

Referenced by `drawKnob()`.

6.30 QwtLegend Class Reference

6.30.1 Detailed Description

The legend widget.

The [QwtLegend](#) widget is a tabular arrangement of legend items. Legend items might be any type of widget, but in general they will be a [QwtLegendItem](#).

See also:

[QwtLegendItem](#), [QwtLegendItemManager](#) [QwtPlot](#)

Definition at line 36 of file qwt_legend.h.

Public Types

- enum [LegendDisplayPolicy](#) {
 NoIdentifier = 0,
 FixedIdentifier = 1,
 AutoIdentifier = 2 }
- enum [LegendItemMode](#) {
 ReadOnlyItem,
 ClickableItem,
 CheckableItem }

Public Member Functions

- [QwtLegend](#) (QWidget *parent=NULL)
- virtual [~QwtLegend](#) ()
- void [setDisplayPolicy](#) ([LegendDisplayPolicy](#) policy, int mode)
- [LegendDisplayPolicy](#) [displayPolicy](#) () const
- void [setItemMode](#) ([LegendItemMode](#))
- [LegendItemMode](#) [itemMode](#) () const
- int [identifierMode](#) () const
- QWidget * [contentsWidget](#) ()
- const QWidget * [contentsWidget](#) () const
- void [insert](#) (const [QwtLegendItemManager](#) *, QWidget *)
- void [remove](#) (const [QwtLegendItemManager](#) *)
- QWidget * [find](#) (const [QwtLegendItemManager](#) *) const
- [QwtLegendItemManager](#) * [find](#) (const QWidget *) const
- virtual QList< QWidget * > [legendItems](#) () const
- void [clear](#) ()
- bool [isEmpty](#) () const
- uint [itemCount](#) () const
- virtual bool [eventFilter](#) (QObject *, QEvent *)
- virtual QSize [sizeHint](#) () const
- virtual int [heightForWidth](#) (int w) const
- QScrollBar * [horizontalScrollBar](#) () const
- QScrollBar * [verticalScrollBar](#) () const

Protected Member Functions

- virtual void [resizeEvent](#) (QResizeEvent *)
- virtual void [layoutContents](#) ()

6.30.2 Member Enumeration Documentation**6.30.2.1 enum [QwtLegend::LegendDisplayPolicy](#)**

Display policy.

- NoIdentifier
The client code is responsible how to display of each legend item. The Qwt library will not interfere.
- FixedIdentifier
All legend items are displayed with the [QwtLegendItem::IdentifierMode](#) to be passed in 'mode'.
- AutoIdentifier
Each legend item is displayed with a mode that is a bitwise or of
 - [QwtLegendItem::ShowLine](#) (if its curve is drawn with a line) and
 - [QwtLegendItem::ShowSymbol](#) (if its curve is drawn with symbols) and
 - [QwtLegendItem::ShowText](#) (if the has a title).

Default is AutoIdentifier.

See also:

[setDisplayPolicy\(\)](#), [displayPolicy\(\)](#), [QwtLegendItem::IdentifierMode](#)

Definition at line 62 of file qwt_legend.h.

6.30.2.2 enum [QwtLegend::LegendItemMode](#)

Interaction mode for the legend items.

Definition at line 70 of file qwt_legend.h.

6.30.3 Constructor & Destructor Documentation

6.30.3.1 [QwtLegend::QwtLegend](#) ([QWidget](#) * *parent* = NULL) [explicit]

Parameters:

parent Parent widget

Definition at line 258 of file qwt_legend.cpp.

6.30.3.2 [QwtLegend::~~QwtLegend](#) () [virtual]

Destructor.

Definition at line 283 of file qwt_legend.cpp.

6.30.4 Member Function Documentation

6.30.4.1 void [QwtLegend::setDisplayPolicy](#) ([LegendDisplayPolicy](#) *policy*, int *mode*)

Set the legend display policy to:

Parameters:

policy Legend display policy

mode Identifier mode (or'd ShowLine, ShowSymbol, ShowText)

See also:

[displayPolicy](#), [LegendDisplayPolicy](#)

Definition at line 296 of file qwt_legend.cpp.

References [QwtLegendItemManager::updateLegend\(\)](#).

6.30.4.2 [QwtLegend::LegendDisplayPolicy](#) [QwtLegend::displayPolicy](#) () const

Returns:

the legend display policy. Default is [LegendDisplayPolicy::Auto](#).

See also:

[setDisplayPolicy](#), [LegendDisplayPolicy](#)

Definition at line 324 of file qwt_legend.cpp.

Referenced by `QwtPlotCurve::updateLegend()`.

6.30.4.3 `int QwtLegend::identifierMode () const`

Returns:

the IdentifierMode to be used in combination with `LegendDisplayPolicy::Fixed`.

Default is `ShowLine | ShowSymbol | ShowText`.

Definition at line 346 of file qwt_legend.cpp.

Referenced by `QwtPlotCurve::updateLegend()`.

6.30.4.4 `QWidget * QwtLegend::contentsWidget ()`

The contents widget is the only child of the viewport() and the parent widget of all legend items.

Definition at line 355 of file qwt_legend.cpp.

Referenced by `insert()`, and `QwtPlot::printLegend()`.

6.30.4.5 `const QWidget * QwtLegend::contentsWidget () const`

The contents widget is the only child of the viewport() and the parent widget of all legend items.

Definition at line 375 of file qwt_legend.cpp.

6.30.4.6 `void QwtLegend::insert (const QwtLegendItemManager * plotItem, QWidget * legendItem)`

Insert a new item for a plot item

Parameters:

plotItem Plot item

legendItem New legend item

Note:

The parent of item will be changed to `QwtLegend::contentsWidget()`

Definition at line 386 of file qwt_legend.cpp.

References `contentsWidget()`, and `layoutContents()`.

Referenced by `QwtPlotItem::updateLegend()`.

6.30.4.7 `void QwtLegend::remove (const QwtLegendItemManager * plotItem)`

Find the corresponding item for a plotItem and remove it from the item list.

Parameters:

plotItem Plot item

Definition at line 482 of file qwt_legend.cpp.

6.30.4.8 QWidget * QwtLegend::find (const QwtLegendItemManager * *plotItem*) const

Find the widget that represents a plot item

Parameters:

plotItem Plot item

Returns:

Widget on the legend, or NULL

Definition at line 460 of file qwt_legend.cpp.

Referenced by QwtPlotPrintFilter::reset(), QwtPlotItem::updateLegend(), and QwtPlotCurve::updateLegend().

6.30.4.9 QwtLegendItemManager * QwtLegend::find (const QWidget * *legendItem*) const

Find the widget that represents a plot item

Parameters:

plotItem Plot item

Returns:

Widget on the legend, or NULL

Definition at line 471 of file qwt_legend.cpp.

6.30.4.10 void QwtLegend::clear ()

Remove all items.

Definition at line 490 of file qwt_legend.cpp.

6.30.4.11 bool QwtLegend::isEmpty () const

Return true, if there are no legend items.

Definition at line 617 of file qwt_legend.cpp.

Referenced by QwtPlotLayout::activate(), and QwtPlot::print().

6.30.4.12 uint QwtLegend::itemCount () const

Return the number of legend items.

Definition at line 623 of file qwt_legend.cpp.

6.30.4.13 `QSize QwtLegend::sizeHint () const` [virtual]

Return a size hint.

Definition at line 506 of file qwt_legend.cpp.

6.30.4.14 `int QwtLegend::heightForWidth (int width) const` [virtual]**Returns:**

The preferred height, for the width w.

Parameters:

width Width

Definition at line 518 of file qwt_legend.cpp.

6.30.4.15 `void QwtLegend::resizeEvent (QResizeEvent * e)` [protected, virtual]

Resize event

Parameters:

e Event

Definition at line 654 of file qwt_legend.cpp.

6.30.4.16 `void QwtLegend::layoutContents ()` [protected, virtual]

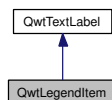
Adjust contents widget and item layout to the size of the viewport().

Definition at line 543 of file qwt_legend.cpp.

Referenced by eventFilter(), and insert().

6.31 QwtLegendItem Class Reference

Inheritance diagram for QwtLegendItem:



Collaboration diagram for QwtLegendItem:



6.31.1 Detailed Description

A legend label.

[QwtLegendItem](#) represents a curve on a legend. It displays an curve identifier with an explaining text. The identifier might be a combination of curve symbol and line. In readonly mode it behaves like a label, otherwise like an unstylish push button.

See also:

[QwtLegend](#), [QwtPlotCurve](#)

Definition at line 35 of file `qwt_legend_item.h`.

Public Types

- enum [IdentifierMode](#) {
 NoIdentifier = 0,
 ShowLine = 1,
 ShowSymbol = 2,
 ShowText = 4 }

Public Slots

- void [setChecked](#) (bool on)

Signals

- void [clicked](#) ()
- void [pressed](#) ()
- void [released](#) ()
- void [checked](#) (bool)

Public Member Functions

- [QwtLegendItem](#) (QWidget *parent=0)
- [QwtLegendItem](#) (const [QwtSymbol](#) &, const QPen &, const [QwtText](#) &, QWidget *parent=0)
- virtual [~QwtLegendItem](#) ()
- virtual void [setText](#) (const [QwtText](#) &)
- void [setItemMode](#) ([QwtLegend::LegendItemMode](#))
- [QwtLegend::LegendItemMode](#) [itemMode](#) () const
- void [setIdentifierMode](#) (int)
- int [identifierMode](#) () const
- void [setIdentifierWidth](#) (int width)
- int [identifierWidth](#) () const
- void [setSpacing](#) (int spacing)
- int [spacing](#) () const
- void [setSymbol](#) (const [QwtSymbol](#) &)
- const [QwtSymbol](#) & [symbol](#) () const

- void [setCurvePen](#) (const QPen &)
- const QPen & [curvePen](#) () const
- virtual void [drawIdentifier](#) (QPainter *, const QRect &) const
- virtual void [drawItem](#) (QPainter *p, const QRect &) const
- virtual QSize [sizeHint](#) () const
- bool [isChecked](#) () const

Protected Member Functions

- void [setDown](#) (bool)
- bool [isDown](#) () const
- virtual void [paintEvent](#) (QPaintEvent *)
- virtual void [mousePressEvent](#) (QMouseEvent *)
- virtual void [mouseReleaseEvent](#) (QMouseEvent *)
- virtual void [keyPressEvent](#) (QKeyEvent *)
- virtual void [keyReleaseEvent](#) (QKeyEvent *)
- virtual void [drawText](#) (QPainter *, const QRect &)

6.31.2 Member Enumeration Documentation

6.31.2.1 enum [QwtLegendItem::IdentifierMode](#)

Identifier mode.

Default is ShowLine | ShowText

See also:

[QwtLegendItem::identifierMode](#), [QwtLegendItem::setIdentifierMode](#)

Definition at line 47 of file `qwt_legend_item.h`.

6.31.3 Constructor & Destructor Documentation

6.31.3.1 [QwtLegendItem::QwtLegendItem](#) (QWidget * *parent* = 0) [explicit]

Parameters:

parent Parent widget

Definition at line 80 of file `qwt_legend_item.cpp`.

6.31.3.2 [QwtLegendItem::QwtLegendItem](#) (const [QwtSymbol](#) & *symbol*, const QPen & *curvePen*, const [QwtText](#) & *text*, QWidget * *parent* = 0) [explicit]

Parameters:

symbol Curve symbol

curvePen Curve pen

text Label text

parent Parent widget

Definition at line 93 of file `qwt_legend_item.cpp`.

References [QwtSymbol::clone\(\)](#), [symbol\(\)](#), and [QwtTextLabel::text\(\)](#).

6.31.3.3 QwtLegendItem::~~QwtLegendItem () [virtual]

Destructor.

Definition at line 116 of file qwt_legend_item.cpp.

6.31.4 Member Function Documentation

6.31.4.1 void QwtLegendItem::setText (const QwtText & text) [virtual]

Set the text to the legend item

Parameters:

text Text label

See also:

[QwtTextLabel::text\(\)](#)

Reimplemented from [QwtTextLabel](#).

Definition at line 128 of file qwt_legend_item.cpp.

References [QwtText::setRenderFlags\(\)](#), [QwtTextLabel::setText\(\)](#), and [QwtTextLabel::text\(\)](#).

6.31.4.2 void QwtLegendItem::setItemMode (QwtLegend::LegendItemMode mode)

Set the item mode The default is [QwtLegend::ReadOnlyItem](#)

Parameters:

mode Item mode

See also:

[itemMode\(\)](#)

Definition at line 150 of file qwt_legend_item.cpp.

References [QwtTextLabel::setMargin\(\)](#).

6.31.4.3 QwtLegend::LegendItemMode QwtLegendItem::itemMode () const

Return the item mode

See also:

[setItemMode\(\)](#)

Definition at line 169 of file qwt_legend_item.cpp.

6.31.4.4 void QwtLegendItem::setIdentifierMode (int mode)

Set identifier mode. Default is [ShowLine](#) | [ShowText](#).

Parameters:

mode Or'd values of [IdentifierMode](#)

See also:

[identifierMode\(\)](#)

Definition at line 181 of file qwt_legend_item.cpp.

6.31.4.5 int QwtLegendItem::identifierMode () const

Or'd values of IdentifierMode.

See also:

[setIdentifierMode\(\)](#), [IdentifierMode](#)

Definition at line 194 of file qwt_legend_item.cpp.

6.31.4.6 void QwtLegendItem::setIdentifierWidth (int *width*)

Set the width for the identifier Default is 8 pixels

Parameters:

width New width

See also:

[identifierMode\(\)](#), [identifierWidth](#)

Definition at line 207 of file qwt_legend_item.cpp.

References [QwtTextLabel::margin\(\)](#), and [QwtTextLabel::setIndent\(\)](#).

6.31.4.7 int QwtLegendItem::identifierWidth () const

Return the width of the identifier

See also:

[setIdentifierWidth](#)

Definition at line 222 of file qwt_legend_item.cpp.

Referenced by [drawItem\(\)](#).

6.31.4.8 void QwtLegendItem::setSpacing (int *spacing*)

Change the spacing

Parameters:

spacing Spacing

See also:

[spacing\(\)](#), [identifierWidth\(\)](#), [QwtTextLabel::margin\(\)](#)

Definition at line 232 of file qwt_legend_item.cpp.

References [QwtTextLabel::margin\(\)](#), and [QwtTextLabel::setIndent\(\)](#).

6.31.4.9 int QwtLegendItem::spacing () const

Return the spacing

See also:

[setSpacing\(\)](#), [identifierWidth\(\)](#), [QwtTextLabel::margin\(\)](#)

Definition at line 247 of file `qwt_legend_item.cpp`.

Referenced by `drawItem()`.

6.31.4.10 void QwtLegendItem::setSymbol (const [QwtSymbol](#) & *symbol*)

Set curve symbol.

Parameters:

symbol Symbol

See also:

[symbol\(\)](#)

Definition at line 258 of file `qwt_legend_item.cpp`.

References `QwtSymbol::clone()`, and `symbol()`.

6.31.4.11 const [QwtSymbol](#) & QwtLegendItem::symbol () const

Returns:

The curve symbol.

See also:

[setSymbol\(\)](#)

Definition at line 269 of file `qwt_legend_item.cpp`.

Referenced by `QwtLegendItem()`, and `setSymbol()`.

6.31.4.12 void QwtLegendItem::setCurvePen (const [QPen](#) & *pen*)

Set curve pen.

Parameters:

pen Curve pen

See also:

[curvePen\(\)](#)

Definition at line 281 of file `qwt_legend_item.cpp`.

6.31.4.13 const QPen & QwtLegendItem::curvePen () const**Returns:**

The curve pen.

See also:

[setCurvePen\(\)](#)

Definition at line 294 of file qwt_legend_item.cpp.

6.31.4.14 void QwtLegendItem::drawIdentifier (QPainter * *painter*, const QRect & *rect*) const
[virtual]

Paint the identifier to a given rect.

Parameters:

painter Painter

rect Rect where to paint

Definition at line 304 of file qwt_legend_item.cpp.

References QwtPainter::drawLine(), QwtPainter::metricsMap(), and QwtMetricsMap::screenToLayout().

Referenced by drawItem(), and paintEvent().

6.31.4.15 void QwtLegendItem::drawItem (QPainter * *painter*, const QRect & *rect*) const
[virtual]

Draw the legend item to a given rect.

Parameters:

painter Painter

rect Rect where to paint the button

Definition at line 360 of file qwt_legend_item.cpp.

References QwtText::draw(), drawIdentifier(), identifierWidth(), QwtTextLabel::margin(), QwtPainter::metricsMap(), QwtMetricsMap::screenToLayoutX(), spacing(), and QwtTextLabel::text().

6.31.4.16 QSize QwtLegendItem::sizeHint () const [virtual]

Return a size hint.

Reimplemented from [QwtTextLabel](#).

Definition at line 578 of file qwt_legend_item.cpp.

References QwtTextLabel::sizeHint().

6.31.4.17 bool QwtLegendItem::isChecked () const

Return true, if the item is checked.

Definition at line 542 of file qwt_legend_item.cpp.

References isDown().

6.31.4.18 void QwtLegendItem::setChecked (bool *on*) [slot]

Check/Uncheck a the item

Parameters:

on check/uncheck

See also:

[setItemMode\(\)](#)

Definition at line 528 of file qwt_legend_item.cpp.

References [setDown\(\)](#).

6.31.4.19 void QwtLegendItem::clicked () [signal]

Signal, when the legend item has been clicked.

Referenced by [setDown\(\)](#).

6.31.4.20 void QwtLegendItem::pressed () [signal]

Signal, when the legend item has been pressed.

Referenced by [setDown\(\)](#).

6.31.4.21 void QwtLegendItem::released () [signal]

Signal, when the legend item has been relased.

Referenced by [setDown\(\)](#).

6.31.4.22 void QwtLegendItem::checked (bool) [signal]

Signal, when the legend item has been toggled.

Referenced by [setDown\(\)](#).

6.31.4.23 void QwtLegendItem::setDown (bool) [protected]

Set the item being down.

Definition at line 548 of file qwt_legend_item.cpp.

References [checked\(\)](#), [clicked\(\)](#), [pressed\(\)](#), and [released\(\)](#).

Referenced by [keyPressEvent\(\)](#), [keyReleaseEvent\(\)](#), [mousePressEvent\(\)](#), [mouseReleaseEvent\(\)](#), and [setChecked\(\)](#).

6.31.4.24 bool QwtLegendItem::isDown () const [protected]

Return true, if the item is down.

Definition at line 572 of file qwt_legend_item.cpp.

Referenced by [isChecked\(\)](#), [keyPressEvent\(\)](#), and [mousePressEvent\(\)](#).

6.31.4.25 void QwtLegendItem::paintEvent (QPaintEvent *) [protected, virtual]

Paint event.

Reimplemented from [QwtTextLabel](#).

Definition at line 385 of file qwt_legend_item.cpp.

References [QwtTextLabel::drawContents\(\)](#), [drawIdentifier\(\)](#), and [QwtTextLabel::margin\(\)](#).

6.31.4.26 void QwtLegendItem::mousePressEvent (QMouseEvent *) [protected, virtual]

Handle mouse press events.

Definition at line 428 of file qwt_legend_item.cpp.

References [isDown\(\)](#), and [setDown\(\)](#).

6.31.4.27 void QwtLegendItem::mouseReleaseEvent (QMouseEvent *) [protected, virtual]

Handle mouse release events.

Definition at line 451 of file qwt_legend_item.cpp.

References [setDown\(\)](#).

6.31.4.28 void QwtLegendItem::keyPressEvent (QKeyEvent *) [protected, virtual]

Handle key press events.

Definition at line 473 of file qwt_legend_item.cpp.

References [isDown\(\)](#), and [setDown\(\)](#).

6.31.4.29 void QwtLegendItem::keyReleaseEvent (QKeyEvent *) [protected, virtual]

Handle key release events.

Definition at line 499 of file qwt_legend_item.cpp.

References [setDown\(\)](#).

6.31.4.30 void QwtLegendItem::drawText (QPainter *, const QRect &) [protected, virtual]

Redraw the text.

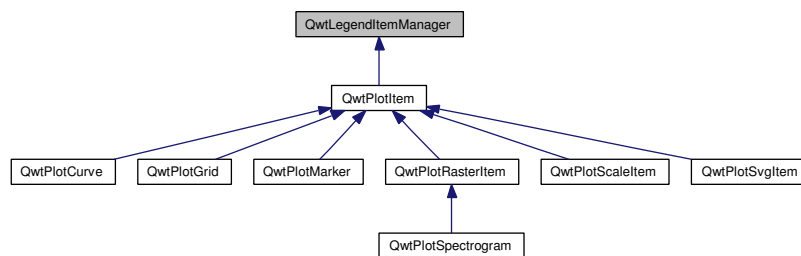
Reimplemented from [QwtTextLabel](#).

Definition at line 587 of file qwt_legend_item.cpp.

References [QwtTextLabel::drawText\(\)](#).

6.32 QwtLegendItemManager Class Reference

Inheritance diagram for QwtLegendItemManager:



6.32.1 Detailed Description

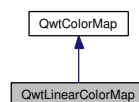
Definition at line 20 of file qwt_legend_itemmanager.h.

Public Member Functions

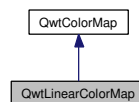
- [QwtLegendItemManager](#) ()
- virtual [~QwtLegendItemManager](#) ()
- virtual void **updateLegend** ([QwtLegend](#) *) const =0
- virtual [QWidget](#) * **legendItem** () const=0

6.33 QwtLinearColorMap Class Reference

Inheritance diagram for QwtLinearColorMap:



Collaboration diagram for QwtLinearColorMap:



6.33.1 Detailed Description

[QwtLinearColorMap](#) builds a color map from color stops.

A color stop is a color at a specific position. The valid range for the positions is [0.0, 1.0]. When mapping a value into a color it is translated into this interval. If [mode\(\)](#) == [FixedColors](#) the color is calculated from the next lower color stop. If [mode\(\)](#) == [ScaledColors](#) the color is calculated by interpolating the colors of the adjacent stops.

Definition at line 111 of file qwt_color_map.h.

Public Types

- enum `Mode` {
`FixedColors`,
`ScaledColors`,
`RotateNeedle`,
`RotateScale` }

Public Member Functions

- `QwtLinearColorMap` (`QwtColorMap::Format`=`QwtColorMap::RGB`)
- `QwtLinearColorMap` (const `QColor` &from, const `QColor` &to, `QwtColorMap::Format`=`QwtColorMap::RGB`)
- `QwtLinearColorMap` (const `QwtLinearColorMap` &)
- virtual `~QwtLinearColorMap` ()
- `QwtLinearColorMap` & `operator=` (const `QwtLinearColorMap` &)
- virtual `QwtColorMap` * `copy` () const
- void `setMode` (`Mode`)
- `Mode` `mode` () const
- void `setColorInterval` (const `QColor` &color1, const `QColor` &color2)
- void `addColorStop` (double value, const `QColor` &)
- `QwtArray`< double > `colorStops` () const
- `QColor` `color1` () const
- `QColor` `color2` () const
- virtual `QRgb` `rgb` (const `QwtDoubleInterval` &, double value) const
- virtual unsigned char `colorIndex` (const `QwtDoubleInterval` &, double value) const

6.33.2 Member Enumeration Documentation**6.33.2.1 enum `QwtLinearColorMap::Mode`**

Mode of color map

See also:

`setMode()`, `mode()`

Definition at line 118 of file `qwt_color_map.h`.

6.33.3 Constructor & Destructor Documentation**6.33.3.1 `QwtLinearColorMap::QwtLinearColorMap` (`QwtColorMap::Format` *format* = `QwtColorMap::RGB`)**

Build a color map with two stops at 0.0 and 1.0. The color at 0.0 is `Qt::blue`, at 1.0 it is `Qt::yellow`.

Parameters:

format Preferred format of the color map

Definition at line 211 of file `qwt_color_map.cpp`.

References `setColorInterval()`.

Referenced by `copy()`.

6.33.3.2 QwtLinearColorMap::QwtLinearColorMap (const QColor & *color1*, const QColor & *color2*, [QwtColorMap::Format](#) *format* = QwtColorMap::RGB)

Build a color map with two stops at 0.0 and 1.0.

Parameters:

color1 Color used for the minimum value of the value interval

color2 Color used for the maximum value of the value interval

format Preferred format of the color map

Definition at line 235 of file qwt_color_map.cpp.

References [setColorInterval\(\)](#).

6.33.3.3 QwtLinearColorMap::QwtLinearColorMap (const [QwtLinearColorMap](#) &)

Copy constructor.

Definition at line 221 of file qwt_color_map.cpp.

6.33.3.4 QwtLinearColorMap::~QwtLinearColorMap () [virtual]

Destructor.

Definition at line 245 of file qwt_color_map.cpp.

6.33.4 Member Function Documentation

6.33.4.1 [QwtLinearColorMap](#) & QwtLinearColorMap::operator= (const [QwtLinearColorMap](#) &)

Assignment operator.

Definition at line 251 of file qwt_color_map.cpp.

References [d_data](#).

6.33.4.2 [QwtColorMap](#) * QwtLinearColorMap::copy () const [virtual]

Clone the color map.

Implements [QwtColorMap](#).

Definition at line 260 of file qwt_color_map.cpp.

References [QwtLinearColorMap\(\)](#).

6.33.4.3 void QwtLinearColorMap::setMode ([Mode](#) *mode*)

Set the mode of the color map.

FixedColors means the color is calculated from the next lower color stop. ScaledColors means the color is calculated by interpolating the colors of the adjacent stops.

See also:

[mode\(\)](#)

Definition at line 277 of file qwt_color_map.cpp.

6.33.4.4 [QwtLinearColorMap::Mode](#) QwtLinearColorMap::mode () const

Returns:

Mode of the color map

See also:

[setMode\(\)](#)

Definition at line 286 of file qwt_color_map.cpp.

6.33.4.5 void QwtLinearColorMap::setColorInterval (const QColor & *color1*, const QColor & *color2*)

Set the color range

Add stops at 0.0 and 1.0.

Parameters:

color1 Color used for the minimum value of the value interval

color2 Color used for the maximum value of the value interval

See also:

[color1\(\)](#), [color2\(\)](#)

Definition at line 301 of file qwt_color_map.cpp.

Referenced by QwtLinearColorMap().

6.33.4.6 void QwtLinearColorMap::addColorStop (double *value*, const QColor & *color*)

Add a color stop

The value has to be in the range [0.0, 1.0]. F.e. a stop at position 17.0 for a range [10.0,20.0] must be passed as: (17.0 - 10.0) / (20.0 - 10.0)

Parameters:

value Value between [0.0, 1.0]

color Color stop

Definition at line 319 of file qwt_color_map.cpp.

6.33.4.7 [QwtArray< double >](#) QwtLinearColorMap::colorStops () const

Return all positions of color stops in increasing order

Definition at line 328 of file qwt_color_map.cpp.

6.33.4.8 QColor QwtLinearColorMap::color1 () const

Returns:

the first color of the color range

See also:

[setColorInterval\(\)](#)

Definition at line 337 of file qwt_color_map.cpp.

6.33.4.9 QColor QwtLinearColorMap::color2 () const

Returns:

the second color of the color range

See also:

[setColorInterval\(\)](#)

Definition at line 346 of file qwt_color_map.cpp.

6.33.4.10 QRgb QwtLinearColorMap::rgb (const [QwtDoubleInterval](#) & *interval*, double *value*) const [virtual]

Map a value of a given interval into a rgb value

Parameters:

interval Range for all values

value Value to map into a rgb value

Implements [QwtColorMap](#).

Definition at line 357 of file qwt_color_map.cpp.

References [QwtDoubleInterval::minValue\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.33.4.11 unsigned char QwtLinearColorMap::colorIndex (const [QwtDoubleInterval](#) & *interval*, double *value*) const [virtual]

Map a value of a given interval into a color index, between 0 and 255

Parameters:

interval Range for all values

value Value to map into a color index

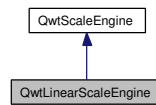
Implements [QwtColorMap](#).

Definition at line 375 of file qwt_color_map.cpp.

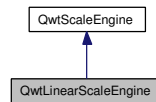
References [QwtDoubleInterval::maxValue\(\)](#), [QwtDoubleInterval::minValue\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.34 QwtLinearScaleEngine Class Reference

Inheritance diagram for QwtLinearScaleEngine:



Collaboration diagram for QwtLinearScaleEngine:



6.34.1 Detailed Description

A scale engine for linear scales.

The step size will fit into the pattern $\{1, 2, 5\} \cdot 10^n$, where n is an integer.

Definition at line 126 of file `qwt_scale_engine.h`.

Public Member Functions

- virtual void [autoScale](#) (int maxSteps, double &x1, double &x2, double &stepSize) const
- virtual [QwtScaleDiv](#) [divideScale](#) (double x1, double x2, int numMajorSteps, int numMinorSteps, double stepSize=0.0) const
- virtual [QwtScaleTransformation](#) * [transformation](#) () const

Protected Member Functions

- [QwtDoubleInterval](#) [align](#) (const [QwtDoubleInterval](#) &, double stepSize) const

6.34.2 Member Function Documentation

6.34.2.1 void [QwtLinearScaleEngine::autoScale](#) (int *maxNumSteps*, double &*x1*, double &*x2*, double &*stepSize*) const [virtual]

Align and divide an interval

Parameters:

- maxNumSteps* Max. number of steps
- x1* First limit of the interval (In/Out)
- x2* Second limit of the interval (In/Out)
- stepSize* Step size (Out)

See also:

[QwtLinearScaleEngine::setAttribute](#)

Implements [QwtScaleEngine](#).

Definition at line 416 of file qwt_scale_engine.cpp.

References [align\(\)](#), [QwtScaleEngine::buildInterval\(\)](#), [QwtScaleEngine::divideInterval\(\)](#), [QwtDoubleInterval::extend\(\)](#), [QwtScaleEngine::hiMargin\(\)](#), [QwtScaleEngine::loMargin\(\)](#), [QwtDoubleInterval::maxValue\(\)](#), [QwtDoubleInterval::minValue\(\)](#), [QwtDoubleInterval::normalized\(\)](#), [QwtScaleEngine::reference\(\)](#), [QwtDoubleInterval::setMaxValue\(\)](#), [QwtDoubleInterval::setMinValue\(\)](#), [QwtDoubleInterval::symmetrize\(\)](#), [QwtScaleEngine::testAttribute\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.34.2.2 [QwtScaleDiv](#) [QwtLinearScaleEngine::divideScale](#) (double *x1*, double *x2*, int *maxMajSteps*, int *maxMinSteps*, double *stepSize* = 0.0) const [virtual]

Calculate a scale division.

Parameters:

x1 First interval limit

x2 Second interval limit

maxMajSteps Maximum for the number of major steps

maxMinSteps Maximum number of minor steps

stepSize Step size. If *stepSize* == 0, the *scaleEngine* calculates one.

See also:

[QwtScaleEngine::stepSize](#), [QwtScaleEngine::subDivide](#)

Implements [QwtScaleEngine](#).

Definition at line 461 of file qwt_scale_engine.cpp.

References [QwtScaleEngine::divideInterval\(\)](#), and [QwtDoubleInterval::width\(\)](#).

Referenced by [QwtLog10ScaleEngine::divideScale\(\)](#), and [QwtDial::updateScale\(\)](#).

6.34.2.3 [QwtScaleTransformation](#) * [QwtLinearScaleEngine::transformation](#) () const [virtual]

Return a transformation, for linear scales

Implements [QwtScaleEngine](#).

Definition at line 401 of file qwt_scale_engine.cpp.

Referenced by [QwtDial::updateScale\(\)](#).

6.34.2.4 [QwtDoubleInterval](#) [QwtLinearScaleEngine::align](#) (const [QwtDoubleInterval](#) & *interval*, double *stepSize*) const [protected]

Align an interval to a step size.

The limits of an interval are aligned that both are integer multiples of the step size.

Parameters:

interval Interval

stepSize Step size

Returns:

Aligned interval

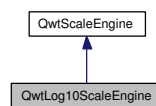
Definition at line 600 of file qwt_scale_engine.cpp.

References `QwtScaleArithmetic::ceilEps()`, `QwtScaleArithmetic::floorEps()`, `QwtDoubleInterval::maxValue()`, and `QwtDoubleInterval::minValue()`.

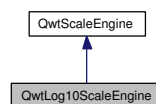
Referenced by `autoScale()`.

6.35 QwtLog10ScaleEngine Class Reference

Inheritance diagram for QwtLog10ScaleEngine:



Collaboration diagram for QwtLog10ScaleEngine:

**6.35.1 Detailed Description**

A scale engine for logarithmic (base 10) scales.

The step size is measured in **decades** and the major step size will be adjusted to fit the pattern $\{1, 2, 3, 5\} \cdot 10^n$, where n is a natural number including zero.

Warning:

the step size as well as the margins are measured in **decades**.

Definition at line 167 of file qwt_scale_engine.h.

Public Member Functions

- virtual void [autoScale](#) (int maxSteps, double &x1, double &x2, double &stepSize) const
- virtual [QwtScaleDiv](#) [divideScale](#) (double x1, double x2, int numMajorSteps, int numMinorSteps, double stepSize=0.0) const
- virtual [QwtScaleTransformation](#) * [transformation](#) () const

Protected Member Functions

- [QwtDoubleInterval](#) [log10](#) (const [QwtDoubleInterval](#) &) const
- [QwtDoubleInterval](#) [pow10](#) (const [QwtDoubleInterval](#) &) const

6.35.2 Member Function Documentation

6.35.2.1 `void QwtLog10ScaleEngine::autoScale (int maxNumSteps, double & x1, double & x2, double & stepSize) const` `[virtual]`

Align and divide an interval

Parameters:

maxNumSteps Max. number of steps
x1 First limit of the interval (In/Out)
x2 Second limit of the interval (In/Out)
stepSize Step size (Out)

See also:

[QwtScaleEngine::setAttribute](#)

Implements [QwtScaleEngine](#).

Definition at line 629 of file `qwt_scale_engine.cpp`.

References [QwtScaleEngine::buildInterval\(\)](#), [QwtScaleEngine::divideInterval\(\)](#), [QwtDoubleInterval::extend\(\)](#), [QwtScaleEngine::hiMargin\(\)](#), [QwtDoubleInterval::limited\(\)](#), [log10\(\)](#), [QwtScaleEngine::loMargin\(\)](#), [QwtDoubleInterval::maxValue\(\)](#), [QwtDoubleInterval::minValue\(\)](#), [QwtScaleEngine::reference\(\)](#), [QwtDoubleInterval::setInterval\(\)](#), [QwtScaleEngine::testAttribute\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.35.2.2 [QwtScaleDiv](#) `QwtLog10ScaleEngine::divideScale (double x1, double x2, int maxMajSteps, int maxMinSteps, double stepSize = 0.0) const` `[virtual]`

Calculate a scale division.

Parameters:

x1 First interval limit
x2 Second interval limit
maxMajSteps Maximum for the number of major steps
maxMinSteps Maximum number of minor steps
stepSize Step size. If `stepSize == 0`, the `scaleEngine` calculates one.

See also:

[QwtScaleEngine::stepSize](#), [QwtLog10ScaleEngine::subDivide](#)

Implements [QwtScaleEngine](#).

Definition at line 686 of file `qwt_scale_engine.cpp`.

References [QwtScaleEngine::attributes\(\)](#), [QwtScaleEngine::divideInterval\(\)](#), [QwtLinearScaleEngine::divideScale\(\)](#), [QwtScaleEngine::hiMargin\(\)](#), [QwtScaleDiv::invert\(\)](#), [QwtDoubleInterval::limited\(\)](#), [log10\(\)](#), [QwtScaleEngine::loMargin\(\)](#), [QwtDoubleInterval::maxValue\(\)](#), [QwtDoubleInterval::minValue\(\)](#), [QwtDoubleInterval::normalized\(\)](#), [QwtScaleEngine::reference\(\)](#), [QwtScaleEngine::setAttribute\(\)](#), [QwtScaleEngine::setMargins\(\)](#), [QwtScaleEngine::setReference\(\)](#), and [QwtDoubleInterval::width\(\)](#).

6.35.2.3 QwtScaleTransformation * QwtLog10ScaleEngine::transformation () const [virtual]

Return a transformation, for logarithmic (base 10) scales

Implements [QwtScaleEngine](#).

Definition at line 614 of file `qwt_scale_engine.cpp`.

6.35.2.4 QwtDoubleInterval QwtLog10ScaleEngine::log10 (const QwtDoubleInterval & interval) const [protected]

Return the interval `[log10(interval.minValue()), log10(interval.maxValue)]`

Definition at line 892 of file `qwt_scale_engine.cpp`.

References `QwtDoubleInterval::maxValue()`, and `QwtDoubleInterval::minValue()`.

Referenced by `autoScale()`, and `divideScale()`.

6.35.2.5 QwtDoubleInterval QwtLog10ScaleEngine::pow10 (const QwtDoubleInterval & interval) const [protected]

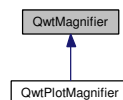
Return the interval `[pow10(interval.minValue()), pow10(interval.maxValue)]`

Definition at line 902 of file `qwt_scale_engine.cpp`.

References `QwtDoubleInterval::maxValue()`, and `QwtDoubleInterval::minValue()`.

6.36 QwtMagnifier Class Reference

Inheritance diagram for QwtMagnifier:

**6.36.1 Detailed Description**

[QwtMagnifier](#) provides zooming, by magnifying in steps.

Using [QwtMagnifier](#) a plot can be zoomed in/out in steps using keys, the mouse wheel or moving a mouse button in vertical direction.

Definition at line 27 of file `qwt_magnifier.h`.

Public Member Functions

- [QwtMagnifier](#) (QWidget *)
- virtual [~QwtMagnifier](#) ()
- QWidget * [parentWidget](#) ()
- const QWidget * [parentWidget](#) () const
- void [setEnabled](#) (bool)
- bool [isEnabled](#) () const

- void [setMouseFactor](#) (double)
- double [mouseFactor](#) () const
- void [setMouseButton](#) (int button, int buttonState=Qt::NoButton)
- void [getMouseButton](#) (int &button, int &buttonState) const
- void [setWheelFactor](#) (double)
- double [wheelFactor](#) () const
- void [setWheelButtonState](#) (int buttonState)
- int [wheelButtonState](#) () const
- void [setKeyFactor](#) (double)
- double [keyFactor](#) () const
- void [setZoomInKey](#) (int key, int modifiers)
- void [getZoomInKey](#) (int &key, int &modifiers) const
- void [setZoomOutKey](#) (int key, int modifiers)
- void [getZoomOutKey](#) (int &key, int &modifiers) const
- virtual bool [eventFilter](#) (QObject *, QEvent *)

Protected Member Functions

- virtual void **rescale** (double factor)=0
- virtual void [widgetMouseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseReleaseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseMoveEvent](#) (QMouseEvent *)
- virtual void [widgetWheelEvent](#) (QWheelEvent *)
- virtual void [widgetKeyPressEvent](#) (QKeyEvent *)
- virtual void [widgetKeyReleaseEvent](#) (QKeyEvent *)

6.36.2 Constructor & Destructor Documentation

6.36.2.1 QwtMagnifier::QwtMagnifier (QWidget * *parent*) [explicit]

Constructor

Parameters:

parent Widget to be magnified

Definition at line 66 of file qwt_magnifier.cpp.

References [setEnabled\(\)](#).

6.36.2.2 QwtMagnifier::~QwtMagnifier () [virtual]

Destructor.

Definition at line 74 of file qwt_magnifier.cpp.

6.36.3 Member Function Documentation

6.36.3.1 void QwtMagnifier::setEnabled (bool *on*)

En/disable the magnifier.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters:

on true or false

See also:

[isEnabled\(\)](#), [eventFilter\(\)](#)

Definition at line 88 of file qwt_magnifier.cpp.

Referenced by QwtMagnifier().

6.36.3.2 bool QwtMagnifier::isEnabled () const**Returns:**

true when enabled, false otherwise

See also:

[setEnabled](#), [eventFilter\(\)](#)

Definition at line 109 of file qwt_magnifier.cpp.

6.36.3.3 void QwtMagnifier::setMouseFactor (double *factor*)

Change the mouse factor.

The mouse factor defines the ratio between the current range on the parent widget and the zoomed range for each vertical mouse movement. The default value is 0.95.

Parameters:

factor Wheel factor

See also:

[mouseFactor\(\)](#), [setMouseButton\(\)](#), [setWheelFactor\(\)](#), [setKeyFactor\(\)](#)

Definition at line 170 of file qwt_magnifier.cpp.

6.36.3.4 double QwtMagnifier::mouseFactor () const**Returns:**

Mouse factor

See also:

[setMouseFactor\(\)](#)

Definition at line 179 of file qwt_magnifier.cpp.

6.36.3.5 void QwtMagnifier::setMouseButton (int *button*, int *buttonState* = Qt::NoButton)

Assign the mouse button, that is used for zooming in/out. The default value is Qt::RightButton.

Parameters:

button Button

buttonState Button state

See also:

[getMouseButton](#)

Definition at line 192 of file qwt_magnifier.cpp.

6.36.3.6 void QwtMagnifier::getMouseButton (int & *button*, int & *buttonState*) const**See also:**

[setMouseButton](#)

Definition at line 199 of file qwt_magnifier.cpp.

6.36.3.7 void QwtMagnifier::setWheelFactor (double *factor*)

Change the wheel factor.

The wheel factor defines the ratio between the current range on the parent widget and the zoomed range for each step of the wheel. The default value is 0.9.

Parameters:

factor Wheel factor

See also:

[wheelFactor\(\)](#), [setWheelButtonState\(\)](#), [setMouseFactor\(\)](#), [setKeyFactor\(\)](#)

Definition at line 125 of file qwt_magnifier.cpp.

6.36.3.8 double QwtMagnifier::wheelFactor () const**Returns:**

Wheel factor

See also:

[setWheelFactor\(\)](#)

Definition at line 134 of file qwt_magnifier.cpp.

6.36.3.9 void QwtMagnifier::setWheelButtonState (int *buttonState*)

Assign a mandatory button state for zooming in/out using the wheel. The default button state is Qt::No-Button.

Parameters:

buttonState Button state

See also:

[wheelButtonState](#)

Definition at line 146 of file qwt_magnifier.cpp.

6.36.3.10 int QwtMagnifier::wheelButtonState () const**Returns:**

Wheel button state

See also:

[setWheelButtonState](#)

Definition at line 155 of file qwt_magnifier.cpp.

6.36.3.11 void QwtMagnifier::setKeyFactor (double *factor*)

Change the key factor.

The key factor defines the ratio between the current range on the parent widget and the zoomed range for each key press of the zoom in/out keys. The default value is 0.9.

Parameters:

factor Key factor

See also:

[keyFactor\(\)](#), [setZoomInKey\(\)](#), [setZoomOutKey\(\)](#), [setWheelFactor](#), [setMouseFactor\(\)](#)

Definition at line 217 of file qwt_magnifier.cpp.

6.36.3.12 double QwtMagnifier::keyFactor () const**Returns:**

Key factor

See also:

[setKeyFactor\(\)](#)

Definition at line 226 of file qwt_magnifier.cpp.

6.36.3.13 void QwtMagnifier::setZoomInKey (int *key*, int *modifiers*)

Assign the key, that is used for zooming in. The default combination is Qt::Key_Plus + Qt::NoModifier.

Parameters:

key
modifiers

See also:

[getZoomInKey\(\)](#), [setZoomOutKey\(\)](#)

Definition at line 239 of file qwt_magnifier.cpp.

6.36.3.14 void QwtMagnifier::getZoomInKey (int & *key*, int & *modifiers*) const**See also:**

[setZoomInKey](#)

Definition at line 246 of file qwt_magnifier.cpp.

6.36.3.15 void QwtMagnifier::setZoomOutKey (int *key*, int *modifiers*)

Assign the key, that is used for zooming out. The default combination is Qt::Key_Minus + Qt::NoModifier.

Parameters:

key
modifiers

See also:

[getZoomOutKey\(\)](#), [setZoomOutKey\(\)](#)

Definition at line 260 of file qwt_magnifier.cpp.

6.36.3.16 void QwtMagnifier::getZoomOutKey (int & *key*, int & *modifiers*) const**See also:**

[setZoomOutKey](#)

Definition at line 267 of file qwt_magnifier.cpp.

6.36.3.17 bool QwtMagnifier::eventFilter (QObject * *o*, QEvent * *e*) [virtual]

Event filter.

When [isEnabled\(\)](#) the mouse events of the observed widget are filtered.

See also:

[widgetMouseEvent\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#) [widgetKeyReleaseEvent\(\)](#)

Definition at line 282 of file qwt_magnifier.cpp.

References `widgetKeyPressEvent()`, `widgetKeyReleaseEvent()`, `widgetMouseMoveEvent()`, `widgetMousePressEvent()`, `widgetMouseReleaseEvent()`, and `widgetWheelEvent()`.

6.36.3.18 `void QwtMagnifier::widgetMousePressEvent (QMouseEvent * me)` [protected, virtual]

Handle a mouse press event for the observed widget.

Parameters:

me Mouse event

See also:

[eventFilter\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#)

Definition at line 330 of file qwt_magnifier.cpp.

References `parentWidget()`.

Referenced by `eventFilter()`.

6.36.3.19 `void QwtMagnifier::widgetMouseReleaseEvent (QMouseEvent *)` [protected, virtual]

Handle a mouse release event for the observed widget.

See also:

[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseMoveEvent\(\)](#),

Definition at line 356 of file qwt_magnifier.cpp.

References `parentWidget()`.

Referenced by `eventFilter()`.

6.36.3.20 `void QwtMagnifier::widgetMouseMoveEvent (QMouseEvent * me)` [protected, virtual]

Handle a mouse move event for the observed widget.

Parameters:

me Mouse event

See also:

[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#),

Definition at line 371 of file qwt_magnifier.cpp.

Referenced by `eventFilter()`.

6.36.3.21 void QwtMagnifier::widgetWheelEvent (QWheelEvent * *we*) [protected, virtual]

Handle a wheel event for the observed widget.

Parameters:

we Wheel event

See also:

[eventFilter\(\)](#)

Definition at line 395 of file qwt_magnifier.cpp.

Referenced by [eventFilter\(\)](#).

6.36.3.22 void QwtMagnifier::widgetKeyPressEvent (QKeyEvent * *ke*) [protected, virtual]

Handle a key press event for the observed widget.

Parameters:

ke Key event

See also:

[eventFilter\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 434 of file qwt_magnifier.cpp.

Referenced by [eventFilter\(\)](#).

6.36.3.23 void QwtMagnifier::widgetKeyReleaseEvent (QKeyEvent *) [protected, virtual]

Handle a key release event for the observed widget.

Parameters:

ke Key event

See also:

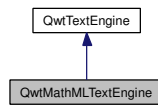
[eventFilter\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 461 of file qwt_magnifier.cpp.

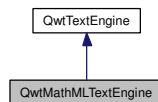
Referenced by [eventFilter\(\)](#).

6.37 QwtMathMLTextEngine Class Reference

Inheritance diagram for QwtMathMLTextEngine:



Collaboration diagram for QwtMathMLTextEngine:



6.37.1 Detailed Description

Text Engine for the MathML renderer of the Qt solutions package.

The Qt Solution package includes a renderer for MathML <http://www.trolltech.com/products/qt/addon/solutions/> that is available for owners of a commercial Qt license. You need a version ≥ 2.1 , that is only available for Qt4.

To enable MathML support the following code needs to be added to the application:

```
#include <qwt_mathml_text_engine.h>

QwtText::setTextEngine(QwtText::MathMLText, new QwtMathMLTextEngine());
```

See also:

[QwtTextEngine](#), [QwtText::setTextEngine](#)

Warning:

Unfortunately the MathML renderer doesn't support rotating of texts.

Definition at line 39 of file `qwt_mathml_text_engine.h`.

Public Member Functions

- [QwtMathMLTextEngine\(\)](#)
- virtual [~QwtMathMLTextEngine\(\)](#)
- virtual int [heightForWidth](#)(const QFont &font, int flags, const QString &text, int width) const
- virtual QSize [textSize](#)(const QFont &font, int flags, const QString &text) const
- virtual void [draw](#)(QPainter *painter, const QRect &rect, int flags, const QString &text) const
- virtual bool [mightRender](#)(const QString &) const
- virtual void [textMargins](#)(const QFont &, const QString &, int &left, int &right, int &top, int &bottom) const

6.37.2 Constructor & Destructor Documentation

6.37.2.1 QwtMathMLTextEngine::QwtMathMLTextEngine()

Constructor.

Definition at line 22 of file `qwt_mathml_text_engine.cpp`.

6.37.2.2 QwtMathMLTextEngine::~QwtMathMLTextEngine () [virtual]

Destructor.

Definition at line 27 of file qwt_mathml_text_engine.cpp.

6.37.3 Member Function Documentation

6.37.3.1 int QwtMathMLTextEngine::heightForWidth (const QFont & *font*, int *flags*, const QString & *text*, int *width*) const [virtual]

Find the height for a given width

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

width Width

Returns:

Calculated height

Implements [QwtTextEngine](#).

Definition at line 41 of file qwt_mathml_text_engine.cpp.

References [textSize\(\)](#).

6.37.3.2 QSize QwtMathMLTextEngine::textSize (const QFont & *font*, int *flags*, const QString & *text*) const [virtual]

Returns the size, that is needed to render text

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

Returns:

Calculated size

Implements [QwtTextEngine](#).

Definition at line 56 of file qwt_mathml_text_engine.cpp.

Referenced by [heightForWidth\(\)](#).

6.37.3.3 void QwtMathMLTextEngine::draw (QPainter * *painter*, const QRect & *rect*, int *flags*, const QString & *text*) const [virtual]

Draw the text in a clipping rectangle

Parameters:

painter Painter
rect Clipping rectangle
flags Bitwise OR of the flags like in for QPainter::drawText
text Text to be rendered

Implements [QwtTextEngine](#).

Definition at line 97 of file qwt_mathml_text_engine.cpp.

6.37.3.4 bool QwtMathMLTextEngine::mightRender (const QString & text) const [virtual]

Test if a string can be rendered by [QwtMathMLTextEngine](#)

Parameters:

text Text to be tested

Returns:

true, if text begins with "<math>".

Implements [QwtTextEngine](#).

Definition at line 131 of file qwt_mathml_text_engine.cpp.

6.37.3.5 void QwtMathMLTextEngine::textMargins (const QFont &, const QString &, int & left, int & right, int & top, int & bottom) const [virtual]

Return margins around the texts

Parameters:

left Return 0
right Return 0
top Return 0
bottom Return 0

Implements [QwtTextEngine](#).

Definition at line 83 of file qwt_mathml_text_engine.cpp.

6.38 QwtMetricsMap Class Reference

6.38.1 Detailed Description

A Map to translate between layout, screen and paint device metrics.

Definition at line 31 of file qwt_layout_metrics.h.

Public Member Functions

- [QwtMetricsMap](#) ()
- bool [isIdentity](#) () const
- void [setMetrics](#) (const QPaintDevice *layoutMetrics, const QPaintDevice *deviceMetrics)
- int [layoutToDeviceX](#) (int x) const
- int [deviceToLayoutX](#) (int x) const
- int [screenToLayoutX](#) (int x) const
- int [layoutToScreenX](#) (int x) const
- int [layoutToDeviceY](#) (int y) const
- int [deviceToLayoutY](#) (int y) const
- int [screenToLayoutY](#) (int y) const
- int [layoutToScreenY](#) (int y) const
- QPoint [layoutToDevice](#) (const QPoint &, const QPainter *=&NULL) const
- QPoint [deviceToLayout](#) (const QPoint &, const QPainter *=&NULL) const
- QPoint [screenToLayout](#) (const QPoint &) const
- QPoint [layoutToScreen](#) (const QPoint &point) const
- QSize [layoutToDevice](#) (const QSize &) const
- QSize [deviceToLayout](#) (const QSize &) const
- QSize [screenToLayout](#) (const QSize &) const
- QSize [layoutToScreen](#) (const QSize &) const
- QRect [layoutToDevice](#) (const QRect &, const QPainter *=&NULL) const
- QRect [deviceToLayout](#) (const QRect &, const QPainter *=&NULL) const
- QRect [screenToLayout](#) (const QRect &) const
- QRect [layoutToScreen](#) (const QRect &) const
- QwtPolygon [layoutToDevice](#) (const QwtPolygon &, const QPainter *=&NULL) const
- QwtPolygon [deviceToLayout](#) (const QwtPolygon &, const QPainter *=&NULL) const

Static Public Member Functions

- static QwtPolygon [translate](#) (const QMatrix &, const QwtPolygon &)
- static QRect [translate](#) (const QMatrix &, const QRect &)

6.38.2 Member Function Documentation**6.38.2.1 QwtPolygon QwtMetricsMap::translate (const QMatrix & *m*, const QwtPolygon & *pa*)**
[static]

Wrapper for QMatrix::map.

Parameters:

m Matrix

pa Polygon to translate

Returns:

Translated polygon

Definition at line 322 of file qwt_layout_metrics.cpp.

Referenced by QwtScaleDraw::labelRect().

6.38.2.2 QRect QwtMetricsMap::translate (const QMatrix & *m*, const QRect & *rect*) [static]

Wrapper for QMatrix::mapRect.

Parameters:

m Matrix

rect Rectangle to translate

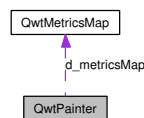
Returns:

Translated rectangle

Definition at line 309 of file qwt_layout_metrics.cpp.

6.39 QwtPainter Class Reference

Collaboration diagram for QwtPainter:



6.39.1 Detailed Description

A collection of QPainter workarounds.

1) Clipping to coordinate system limits (Qt3 only)

On X11 pixel coordinates are stored in shorts. Qt produces overruns when mapping QCOORDS to shorts.

2) Scaling to device metrics

QPainter scales fonts, line and fill patterns to the metrics of the paint device. Other values like the geometries of rects, points remain device independent. To enable a device independent widget implementation, [QwtPainter](#) adds scaling of these geometries. (Unfortunately QPainter::scale scales both types of paintings, so the objects of the first type would be scaled twice).

Definition at line 62 of file qwt_painter.h.

Static Public Member Functions

- static void [setMetricsMap](#) (const QPaintDevice *layout, const QPaintDevice *device)
- static void [setMetricsMap](#) (const [QwtMetricsMap](#) &)
- static void [resetMetricsMap](#) ()
- static const [QwtMetricsMap](#) & [metricsMap](#) ()
- static void [setDeviceClipping](#) (bool)
- static bool [deviceClipping](#) ()
- static void [setClipRect](#) (QPainter *, const QRect &)
- static void [drawText](#) (QPainter *, int x, int y, const QString &)
- static void [drawText](#) (QPainter *, const QPoint &, const QString &)

- static void [drawText](#) (QPainter *, int x, int y, int w, int h, int flags, const QString &)
- static void [drawText](#) (QPainter *, const QRect &, int flags, const QString &)
- static void [drawSimpleRichText](#) (QPainter *, const QRect &, int flags, QTextDocument &)
- static void [drawRect](#) (QPainter *, int x, int y, int w, int h)
- static void [drawRect](#) (QPainter *, const QRect &rect)
- static void [fillRect](#) (QPainter *, const QRect &, const QBrush &)
- static void [drawEllipse](#) (QPainter *, const QRect &)
- static void [drawPie](#) (QPainter *, const QRect &r, int a, int alen)
- static void [drawLine](#) (QPainter *, int x1, int y1, int x2, int y2)
- static void [drawLine](#) (QPainter *, const QPoint &p1, const QPoint &p2)
- static void [drawPolygon](#) (QPainter *, const QwtPolygon &pa)
- static void [drawPolyline](#) (QPainter *, const QwtPolygon &pa)
- static void [drawPoint](#) (QPainter *, int x, int y)
- static void [drawRoundFrame](#) (QPainter *, const QRect &, int width, const QPalette &, bool sunken)
- static void [drawFocusRect](#) (QPainter *, QWidget *)
- static void [drawFocusRect](#) (QPainter *, QWidget *, const QRect &)
- static QwtPolygon [clip](#) (const QwtPolygon &)
- static void [drawColorBar](#) (QPainter *painter, const [QwtColorMap](#) &, const [QwtDoubleInterval](#) &, const [QwtScaleMap](#) &, Qt::Orientation, const QRect &)

6.39.2 Member Function Documentation

6.39.2.1 void QwtPainter::setMetricsMap (const QPaintDevice * *layout*, const QPaintDevice * *device*) [static]

Scale all [QwtPainter](#) drawing operations using the ratio `QwtPaintMetrics(from).logicalDpiX() / QwtPaintMetrics(to).logicalDpiX()` and `QwtPaintMetrics(from).logicalDpiY() / QwtPaintMetrics(to).logicalDpiY()`

See also:

`QwtPainter::resetScaleMetrics()`, `QwtPainter::scaleMetricsX`, `QwtPainter::scaleMetricsY()`

Definition at line 133 of file `qwt_painter.cpp`.

References `QwtMetricsMap::setMetrics()`.

Referenced by `QwtScaleDraw::drawTick()`, and `QwtPlot::print()`.

6.39.2.2 void QwtPainter::setMetricsMap (const [QwtMetricsMap](#) & *map*) [static]

Change the metrics map

See also:

[QwtPainter::resetMetricsMap](#), [QwtPainter::metricsMap](#)

Definition at line 143 of file `qwt_painter.cpp`.

6.39.2.3 void QwtPainter::resetMetricsMap () [static]

Reset the metrics map to the ratio 1:1

See also:

[QwtPainter::setMetricsMap](#), [QwtPainter::resetMetricsMap](#)

Definition at line 152 of file qwt_painter.cpp.

Referenced by QwtScaleDraw::drawTick(), and QwtPlot::print().

6.39.2.4 `const QwtMetricsMap & QwtPainter::metricsMap ()` [static]

Returns:

Metrics map

Definition at line 160 of file qwt_painter.cpp.

Referenced by QwtText::draw(), QwtSymbol::draw(), QwtPlotMarker::draw(), QwtPlotCurve::draw(), QwtLegendItem::drawIdentifier(), QwtLegendItem::drawItem(), QwtPlotCurve::drawSymbols(), QwtScaleDraw::drawTick(), QwtText::heightForWidth(), QwtPlot::print(), QwtPlot::printScale(), and QwtText::textSize().

6.39.2.5 `void QwtPainter::setDeviceClipping (bool enable)` [static]

En/Disable device clipping.

On X11 the default for device clipping is enabled, otherwise it is disabled.

See also:

[QwtPainter::deviceClipping\(\)](#)

Definition at line 62 of file qwt_painter.cpp.

6.39.2.6 `bool QwtPainter::deviceClipping ()` [static]

Returns whether device clipping is enabled. On X11 the default is enabled, otherwise it is disabled.

See also:

[QwtPainter::setDeviceClipping\(\)](#)

Definition at line 73 of file qwt_painter.cpp.

6.39.2.7 `void QwtPainter::setClipRect (QPainter * painter, const QRect & rect)` [static]

Wrapper for QPainter::setClipRect()

Definition at line 168 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

Referenced by QwtPlot::printCanvas(), and QwtPlot::printLegend().

6.39.2.8 `void QwtPainter::drawText (QPainter * painter, int x, int y, const QString & text)` [static]

Wrapper for QPainter::drawText()

Definition at line 310 of file qwt_painter.cpp.

Referenced by QwtPlainTextEngine::draw(), and drawText().

6.39.2.9 `void QwtPainter::drawText (QPainter * painter, const QPoint & pos, const QString & text)` `[static]`

Wrapper for QPainter::drawText()

Definition at line 319 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

6.39.2.10 `void QwtPainter::drawText (QPainter * painter, int x, int y, int w, int h, int flags, const QString & text)` `[static]`

Wrapper for QPainter::drawText()

Definition at line 335 of file qwt_painter.cpp.

References drawText().

6.39.2.11 `void QwtPainter::drawText (QPainter * painter, const QRect & rect, int flags, const QString & text)` `[static]`

Wrapper for QPainter::drawText()

Definition at line 344 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

6.39.2.12 `void QwtPainter::drawSimpleRichText (QPainter * painter, const QRect & rect, int flags, QTextDocument & text)` `[static]`

Wrapper for QSimpleRichText::draw()

Definition at line 391 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

Referenced by QwtRichTextEngine::draw().

6.39.2.13 `void QwtPainter::drawRect (QPainter * painter, int x, int y, int w, int h)` `[static]`

Wrapper for QPainter::drawRect()

Definition at line 176 of file qwt_painter.cpp.

Referenced by QwtText::draw(), QwtSymbol::draw(), QwtPicker::drawRubberBand(), and QwtPlot::printCanvas().

6.39.2.14 `void QwtPainter::drawRect (QPainter * painter, const QRect & rect)` `[static]`

Wrapper for QPainter::drawRect()

Definition at line 184 of file qwt_painter.cpp.

References drawPolyline(), fillRect(), and QwtMetricsMap::layoutToDevice().

6.39.2.15 `void QwtPainter::fillRect (QPainter * painter, const QRect & rect, const QBrush & brush)` `[static]`

Wrapper for QPainter::fillRect()

Definition at line 228 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

Referenced by drawRect(), and QwtPlot::printCanvas().

6.39.2.16 void QwtPainter::drawEllipse (QPainter * *painter*, const QRect & *rect*) [static]

Wrapper for QPainter::drawEllipse()

Definition at line 281 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

Referenced by QwtSymbol::draw(), and QwtPicker::drawRubberBand().

6.39.2.17 void QwtPainter::drawPie (QPainter * *painter*, const QRect & *rect*, int *a*, int *alen*) [static]

Wrapper for QPainter::drawPie()

Definition at line 266 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

6.39.2.18 void QwtPainter::drawLine (QPainter * *painter*, int *x1*, int *y1*, int *x2*, int *y2*) [static]

Wrapper for QPainter::drawLine()

Definition at line 424 of file qwt_painter.cpp.

References drawPolyline(), QwtMetricsMap::isIdentity(), and QwtMetricsMap::layoutToDevice().

Referenced by QwtSymbol::draw(), QwtPlotMarker::draw(), QwtScaleDraw::drawBackbone(), QwtPlotSpectrogram::drawContourLines(), QwtLegendItem::drawIdentifier(), drawLine(), QwtPicker::drawRubberBand(), QwtPlotCurve::drawSticks(), QwtScaleDraw::drawTick(), and QwtRoundScaleDraw::drawTick().

6.39.2.19 void QwtPainter::drawLine (QPainter *, const QPoint & *p1*, const QPoint & *p2*) [inline, static]

Wrapper for QPainter::drawLine().

Definition at line 143 of file qwt_painter.h.

References drawLine().

6.39.2.20 void QwtPainter::drawPolygon (QPainter * *painter*, const QwtPolygon & *pa*) [static]

Wrapper for QPainter::drawPolygon()

Definition at line 474 of file qwt_painter.cpp.

References clip(), and QwtMetricsMap::layoutToDevice().

Referenced by QwtSymbol::draw(), and QwtPlotCurve::fillCurve().

6.39.2.21 void QwtPainter::drawPolyline (QPainter * *painter*, const QwtPolygon & *pa*) [static]

Wrapper for QPainter::drawPolyline()

Definition at line 491 of file qwt_painter.cpp.

References clip(), and QwtMetricsMap::layoutToDevice().

Referenced by drawLine(), QwtPlotCurve::drawLines(), drawRect(), and QwtPlotCurve::drawSteps().

6.39.2.22 void QwtPainter::drawPoint (QPainter *painter, int x, int y) [static]

Wrapper for QPainter::drawPoint()

Definition at line 534 of file qwt_painter.cpp.

References QwtMetricsMap::layoutToDevice().

Referenced by QwtPlotCurve::drawDots().

6.39.2.23 void QwtPainter::drawRoundFrame (QPainter *, const QRect &, int width, const QPalette &, bool sunken) [static]

Draw a round frame.

Definition at line 608 of file qwt_painter.cpp.

Referenced by QwtDial::drawFrame().

6.39.2.24 QwtPolygon QwtPainter::clip (const QwtPolygon &) [static]

Clip a point array.

Definition at line 95 of file qwt_painter.cpp.

References QwtClipper::clipPolygon().

Referenced by drawPolygon(), and drawPolyline().

6.40 QwtPanner Class Reference

Inheritance diagram for QwtPanner:



6.40.1 Detailed Description

[QwtPanner](#) provides panning of a widget.

[QwtPanner](#) grabs the contents of a widget, that can be dragged in all directions. The offset between the start and the end position is emitted by the panned signal.

[QwtPanner](#) grabs the content of the widget into a pixmap and moves the pixmap around, without initiating any repaint events for the widget. Areas, that are not part of content are not painted while panning in process. This makes panning fast enough for widgets, where repaints are too slow for mouse movements.

For widgets, where repaints are very fast it might be better to implement panning manually by mapping mouse events into paint events.

Definition at line 35 of file qwt_panner.h.

Signals

- void [panned](#) (int dx, int dy)
- void [moved](#) (int dx, int dy)

Public Member Functions

- [QwtPanner](#) (QWidget *parent)
- virtual [~QwtPanner](#) ()
- void [setEnabled](#) (bool)
- bool [isEnabled](#) () const
- void [setMouseButton](#) (int button, int buttonState=Qt::NoButton)
- void [getMouseButton](#) (int &button, int &buttonState) const
- void [setAbortKey](#) (int key, int state=Qt::NoButton)
- void [getAbortKey](#) (int &key, int &state) const
- void [setCursor](#) (const QCursor &)
- const QCursor [cursor](#) () const
- void [setOrientations](#) (Qt::Orientations)
- Qt::Orientations [orientations](#) () const
- bool [isOrientationEnabled](#) (Qt::Orientation) const
- virtual bool [eventFilter](#) (QObject *, QEvent *)

Protected Member Functions

- virtual void [widgetMouseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseReleaseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseMoveEvent](#) (QMouseEvent *)
- virtual void [widgetKeyPressEvent](#) (QKeyEvent *)
- virtual void [widgetKeyReleaseEvent](#) (QKeyEvent *)
- virtual void [paintEvent](#) (QPaintEvent *)

6.40.2 Constructor & Destructor Documentation

6.40.2.1 QwtPanner::QwtPanner (QWidget *parent)

Creates an panner that is enabled for the left mouse button.

Parameters:

parent Parent widget to be panned

Definition at line 121 of file qwt_panner.cpp.

References [setEnabled\(\)](#).

6.40.2.2 QwtPanner::~QwtPanner () [virtual]

Destructor.

Definition at line 140 of file qwt_panner.cpp.

6.40.3 Member Function Documentation

6.40.3.1 void QwtPanner::setEnabled (bool *on*)

En/disable the panner.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters:

on true or false

See also:

[isEnabled\(\)](#), [eventFilter\(\)](#)

Definition at line 220 of file qwt_panner.cpp.

Referenced by [QwtPanner\(\)](#), and [widgetMouseEvent\(\)](#).

6.40.3.2 bool QwtPanner::isEnabled () const

Returns:

true when enabled, false otherwise

See also:

[setEnabled](#), [eventFilter\(\)](#)

Definition at line 287 of file qwt_panner.cpp.

6.40.3.3 void QwtPanner::setMouseButton (int *button*, int *buttonState* = Qt::NoButton)

Change the mouse button The defaults are Qt::LeftButton and Qt::NoButton

Definition at line 149 of file qwt_panner.cpp.

6.40.3.4 void QwtPanner::getMouseButton (int & *button*, int & *buttonState*) const

Get the mouse button.

Definition at line 156 of file qwt_panner.cpp.

6.40.3.5 void QwtPanner::setAbortKey (int *key*, int *state* = Qt::NoButton)

Change the abort key The defaults are Qt::Key_Escape and Qt::NoButton

Definition at line 166 of file qwt_panner.cpp.

6.40.3.6 void QwtPanner::getAbortKey (int & *key*, int & *state*) const

Get the abort key.

Definition at line 173 of file qwt_panner.cpp.

6.40.3.7 void QwtPanner::setCursor (const QCursor & *cursor*)

Change the cursor, that is active while panning The default is the cursor of the parent widget.

Parameters:

cursor New cursor

See also:

[setCursor\(\)](#)

Definition at line 188 of file qwt_panner.cpp.

6.40.3.8 const QCursor QwtPanner::cursor () const**Returns:**

Cursor that is active while panning

See also:

[setCursor\(\)](#)

Definition at line 199 of file qwt_panner.cpp.

6.40.3.9 void QwtPanner::setOrientations (Qt::Orientations *o*)

Set the orientations, where panning is enabled The default value is in both directions: Qt::Horizontal | Qt::Vertical

/param *o* Orientation

Definition at line 249 of file qwt_panner.cpp.

6.40.3.10 Qt::Orientations QwtPanner::orientations () const

Return the orientation, where panning is enabled.

Definition at line 255 of file qwt_panner.cpp.

6.40.3.11 bool QwtPanner::isOrientationEnabled (Qt::Orientation *o*) const

Return true if a orientation is enabled

See also:

[orientations\(\)](#), [setOrientations\(\)](#)

Definition at line 272 of file qwt_panner.cpp.

Referenced by widgetMouseMoveEvent(), and widgetMouseReleaseEvent().

6.40.3.12 bool QwtPanner::eventFilter (QObject * *o*, QEvent * *e*) [virtual]

Event filter.

When [isEnabled\(\)](#) the mouse events of the observed widget are filtered.

See also:

[widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#)

Definition at line 341 of file qwt_panner.cpp.

References [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetMousePressEvent\(\)](#), and [widgetMouseReleaseEvent\(\)](#).

6.40.3.13 void QwtPanner::panned (int *dx*, int *dy*) [signal]

Signal emitted, when panning is done

Parameters:

dx Offset in horizontal direction

dy Offset in vertical direction

Referenced by [QwtPlotPanner::QwtPlotPanner\(\)](#), and [widgetMouseReleaseEvent\(\)](#).

6.40.3.14 void QwtPanner::moved (int *dx*, int *dy*) [signal]

Signal emitted, while the widget moved, but panning is not finished.

Parameters:

dx Offset in horizontal direction

dy Offset in vertical direction

Referenced by [widgetMouseMoveEvent\(\)](#).

6.40.3.15 void QwtPanner::widgetMousePressEvent (QMouseEvent * *me*) [protected, virtual]

Handle a mouse press event for the observed widget.

Parameters:

me Mouse event

See also:

[eventFilter\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#),

Definition at line 392 of file qwt_panner.cpp.

References [setEnabled\(\)](#).

Referenced by [eventFilter\(\)](#).

6.40.3.16 void QwtPanner::widgetMouseReleaseEvent (QMouseEvent * *me*) [protected, virtual]

Handle a mouse release event for the observed widget.

Parameters:

me Mouse event

See also:

[eventFilter\(\)](#), [widgetMouseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#),

Definition at line 474 of file qwt_panner.cpp.

References [isOrientationEnabled\(\)](#), and [panned\(\)](#).

Referenced by [eventFilter\(\)](#).

6.40.3.17 void QwtPanner::widgetMouseMoveEvent (QMouseEvent * *me*) [protected, virtual]

Handle a mouse move event for the observed widget.

Parameters:

me Mouse event

See also:

[eventFilter\(\)](#), [widgetMouseEvent\(\)](#), [widgetMousePressEvent\(\)](#)

Definition at line 446 of file qwt_panner.cpp.

References [isOrientationEnabled\(\)](#), and [moved\(\)](#).

Referenced by [eventFilter\(\)](#).

6.40.3.18 void QwtPanner::widgetKeyPressEvent (QKeyEvent * *ke*) [protected, virtual]

Handle a key press event for the observed widget.

Parameters:

ke Key event

See also:

[eventFilter\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 506 of file qwt_panner.cpp.

Referenced by [eventFilter\(\)](#).

6.40.3.19 void QwtPanner::widgetKeyReleaseEvent (QKeyEvent *) [protected, virtual]

Handle a key release event for the observed widget.

Parameters:

ke Key event

See also:

[eventFilter\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 535 of file qwt_panner.cpp.

Referenced by [eventFilter\(\)](#).

6.40.3.20 void QwtPanner::paintEvent (QPaintEvent **pe*) [protected, virtual]

Paint event.

Repaint the grabbed pixmap on its current position and fill the empty spaces by the background of the parent widget.

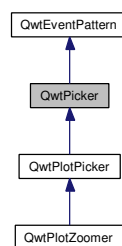
Parameters:

pe Paint event

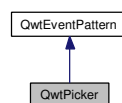
Definition at line 300 of file qwt_panner.cpp.

6.41 QwtPicker Class Reference

Inheritance diagram for QwtPicker:



Collaboration diagram for QwtPicker:



6.41.1 Detailed Description

[QwtPicker](#) provides selections on a widget.

[QwtPicker](#) filters all mouse and keyboard events of a widget and translates them into an array of selected points. Depending on the [QwtPicker::SelectionType](#) the selection might be a single point, a rectangle or a polygon. The selection process is supported by optional rubberbands (rubberband selection) and position trackers.

[QwtPicker](#) is useful for widgets where the event handlers can't be overloaded, like for components of composite widgets. It offers alternative handlers for mouse and key events.

Example

```

#include <qwt_picker.h>

QwtPicker *picker = new QwtPicker(widget);
picker->setTrackerMode(QwtPicker::ActiveOnly);
connect(picker, SIGNAL(selected(const QwtPolygon &)), ...);

// emit the position of clicks on widget
picker->setSelectionFlags(QwtPicker::PointSelection | QwtPicker::ClickSelection);

...

// now select rectangles
picker->setSelectionFlags(QwtPicker::RectSelection | QwtPicker::DragSelection);
picker->setRubberBand(QwtPicker::RectRubberBand);

```

The selection process uses the commands [begin\(\)](#), [append\(\)](#), [move\(\)](#) and [end\(\)](#). [append\(\)](#) adds a new point to the selection, [move\(\)](#) changes the position of the latest point.

The commands are initiated from a small state machine ([QwtPickerMachine](#)) that translates mouse and key events. There are a couple of predefined state machines for point, rect and polygon selections. The [selectionFlags\(\)](#) control which one should be used. It is possible to use other machines by overloading [stateMachine\(\)](#).

The picker is active ([isActive\(\)](#)), between [begin\(\)](#) and [end\(\)](#). In active state the rubberband is displayed, and the tracker is visible in case of trackerMode is ActiveOnly or AlwaysOn.

The cursor can be moved using the arrow keys. All selections can be aborted using the abort key. ([Qwt-EventPattern::KeyPatternCode](#))

Warning:

In case of QWidget::NoFocus the focus policy of the observed widget is set to QWidget::WheelFocus and mouse tracking will be manipulated for ClickSelection while the picker is active, or if [trackerMode\(\)](#) is AlwaysOn.

Definition at line 80 of file qwt_picker.h.

Public Types

- enum [SelectionType](#) {
NoSelection = 0,
PointSelection = 1,
RectSelection = 2,
PolygonSelection = 4 }
- enum [RectSelectionType](#) {
CornerToCorner = 64,
CenterToCorner = 128,
CenterToRadius = 256 }
- enum [SelectionMode](#) {
ClickSelection = 1024,
DragSelection = 2048 }

- enum [RubberBand](#) {
 NoRubberBand = 0,
 HLineRubberBand,
 VLineRubberBand,
 CrossRubberBand,
 RectRubberBand,
 EllipseRubberBand,
 PolygonRubberBand,
 UserRubberBand = 100 }
- enum [DisplayMode](#) {
 AlwaysOff,
 AlwaysOn,
 ActiveOnly,
 ImageMode = 1,
 ContourMode = 2 }
- enum [ResizeMode](#) {
 Stretch,
 KeepSize }

Signals

- void [selected](#) (const QwtPolygon &pa)
- void [appended](#) (const QPoint &pos)
- void [moved](#) (const QPoint &pos)
- void [changed](#) (const QwtPolygon &pa)

Public Member Functions

- [QwtPicker](#) (QWidget *parent)
- [QwtPicker](#) (int selectionFlags, [RubberBand](#) rubberBand, [DisplayMode](#) trackerMode, QWidget *)
- virtual [~QwtPicker](#) ()
- virtual void [setSelectionFlags](#) (int)
- int [selectionFlags](#) () const
- virtual void [setRubberBand](#) ([RubberBand](#))
- [RubberBand](#) rubberBand () const
- virtual void [setTrackerMode](#) ([DisplayMode](#))
- [DisplayMode](#) trackerMode () const
- virtual void [setResizeMode](#) ([ResizeMode](#))
- [ResizeMode](#) resizeMode () const
- virtual void [setRubberBandPen](#) (const QPen &)
- QPen [rubberBandPen](#) () const
- virtual void [setTrackerPen](#) (const QPen &)
- QPen [trackerPen](#) () const
- virtual void [setTrackerFont](#) (const QFont &)
- QFont [trackerFont](#) () const
- bool [isEnabled](#) () const

- virtual void [setEnabled](#) (bool)
- bool [isActive](#) () const
- virtual bool [eventFilter](#) (QObject *, QEvent *)
- QWidget * [parentWidget](#) ()
- const QWidget * [parentWidget](#) () const
- virtual QRect [pickRect](#) () const
- const QwtPolygon & [selection](#) () const
- virtual void [drawRubberBand](#) (QPainter *) const
- virtual void [drawTracker](#) (QPainter *) const
- virtual [QwtText](#) [trackerText](#) (const QPoint &pos) const
- QPoint [trackerPosition](#) () const
- QRect [trackerRect](#) (const QFont &) const

Protected Member Functions

- virtual bool [accept](#) (QwtPolygon &selection) const
- virtual void [transition](#) (const QEvent *)
- virtual void [begin](#) ()
- virtual void [append](#) (const QPoint &)
- virtual void [move](#) (const QPoint &)
- virtual bool [end](#) (bool ok=true)
- virtual void [reset](#) ()
- virtual void [widgetMouseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseReleaseEvent](#) (QMouseEvent *)
- virtual void [widgetMouseDoubleClickEvent](#) (QMouseEvent *)
- virtual void [widgetMouseMoveEvent](#) (QMouseEvent *)
- virtual void [widgetWheelEvent](#) (QWheelEvent *)
- virtual void [widgetKeyPressEvent](#) (QKeyEvent *)
- virtual void [widgetKeyReleaseEvent](#) (QKeyEvent *)
- virtual void [widgetLeaveEvent](#) (QEvent *)
- virtual void [stretchSelection](#) (const QSize &oldSize, const QSize &newSize)
- virtual [QwtPickerMachine](#) * [stateMachine](#) (int) const
- virtual void [updateDisplay](#) ()
- const QWidget * [rubberBandWidget](#) () const
- const QWidget * [trackerWidget](#) () const

6.41.2 Member Enumeration Documentation

6.41.2.1 enum [QwtPicker::SelectionType](#)

This enum type describes the type of a selection. It can be or'd with [QwtPicker::RectSelectionType](#) and [QwtPicker::SelectionMode](#) and passed to [QwtPicker::setSelectionFlags\(\)](#)

- NoSelection
Selection is disabled. Note this is different to the disabled state, as you might have a tracker.
- PointSelection
Select a single point.
- RectSelection
Select a rectangle.

- PolygonSelection
Select a polygon.

The default value is NoSelection.

See also:

[QwtPicker::setSelectionFlags\(\)](#), [QwtPicker::selectionFlags\(\)](#)

Definition at line 117 of file qwt_picker.h.

6.41.2.2 enum [QwtPicker::RectSelectionType](#)

Selection subtype for RectSelection This enum type describes the type of rectangle selections. It can be or'd with [QwtPicker::RectSelectionType](#) and [QwtPicker::SelectionMode](#) and passed to [QwtPicker::setSelectionFlags\(\)](#).

- CornerToCorner
The first and the second selected point are the corners of the rectangle.
- CenterToCorner
The first point is the center, the second a corner of the rectangle.
- CenterToRadius
The first point is the center of a quadrat, calculated by the maximum of the x- and y-distance.

The default value is CornerToCorner.

See also:

[QwtPicker::setSelectionFlags\(\)](#), [QwtPicker::selectionFlags\(\)](#)

Definition at line 143 of file qwt_picker.h.

6.41.2.3 enum [QwtPicker::SelectionMode](#)

Values of this enum type or'd together with a SelectionType value identifies which state machine should be used for the selection.

The default value is ClickSelection.

See also:

[stateMachine\(\)](#)

Definition at line 157 of file qwt_picker.h.

6.41.2.4 enum [QwtPicker::RubberBand](#)

Rubberband style

- NoRubberBand
No rubberband.

- `HLineRubberBand` & `PointSelection`
A horizontal line.
- `VLineRubberBand` & `PointSelection`
A vertical line.
- `CrossRubberBand` & `PointSelection`
A horizontal and a vertical line.
- `RectRubberBand` & `RectSelection`
A rectangle.
- `EllipseRubberBand` & `RectSelection`
An ellipse.
- `PolygonRubberBand` & `PolygonSelection`
A polygon.
- `UserRubberBand`
Values \geq `UserRubberBand` can be used to define additional rubber bands.

The default value is `NoRubberBand`.

See also:

[QwtPicker::setRubberBand\(\)](#), [QwtPicker::rubberBand\(\)](#)

Definition at line 187 of file `qwt_picker.h`.

6.41.2.5 enum `QwtPicker::DisplayMode`

- `AlwaysOff`
Display never.
- `AlwaysOn`
Display always.
- `ActiveOnly`
Display only when the selection is active.

See also:

[QwtPicker::setTrackerMode\(\)](#), [QwtPicker::trackerMode\(\)](#), [QwtPicker::isActive\(\)](#)

Definition at line 217 of file `qwt_picker.h`.

6.41.2.6 enum [QwtPicker::ResizeMode](#)

Controls what to do with the selected points of an active selection when the observed widget is resized.

- **Stretch**
All points are scaled according to the new size,
- **KeepSize**
All points remain unchanged.

The default value is **Stretch**.

See also:

[QwtPicker::setResizeMode\(\)](#), [QwtPicker::resize\(\)](#)

Definition at line 236 of file `qwt_picker.h`.

6.41.3 Constructor & Destructor Documentation

6.41.3.1 [QwtPicker::QwtPicker \(QWidget *parent\)](#) `[explicit]`

Constructor

Creates an picker that is enabled, but where selection flag is set to **NoSelection**, rubberband and tracker are disabled.

Parameters:

parent Parent widget, that will be observed

Definition at line 249 of file `qwt_picker.cpp`.

6.41.3.2 [QwtPicker::QwtPicker \(int selectionFlags, \[RubberBand\]\(#\) rubberBand, \[DisplayMode\]\(#\) trackerMode, QWidget *parent\)](#) `[explicit]`

Constructor

Parameters:

selectionFlags Or'd value of **SelectionType**, **RectSelectionType** and **SelectionMode**

rubberBand Rubberband style

trackerMode Tracker mode

parent Parent widget, that will be observed

Definition at line 264 of file `qwt_picker.cpp`.

6.41.3.3 [QwtPicker::~QwtPicker \(\)](#) `[virtual]`

Destructor.

Definition at line 272 of file `qwt_picker.cpp`.

6.41.4 Member Function Documentation

6.41.4.1 void QwtPicker::setSelectionFlags (int *flags*) [virtual]

Set the selection flags

Parameters:

flags Or'd value of SelectionType, RectSelectionType and SelectionMode. The default value is No-Selection.

See also:

[selectionFlags\(\)](#), [SelectionType](#), [RectSelectionType](#), [SelectionMode](#)

Reimplemented in [QwtPlotZoomer](#).

Definition at line 403 of file qwt_picker.cpp.

References [stateMachine\(\)](#).

Referenced by [QwtPlotZoomer::setSelectionFlags\(\)](#).

6.41.4.2 int QwtPicker::selectionFlags () const

Returns:

Selection flags, an Or'd value of SelectionType, RectSelectionType and SelectionMode.

See also:

[setSelectionFlags\(\)](#), [SelectionType](#), [RectSelectionType](#), [SelectionMode](#)

Definition at line 414 of file qwt_picker.cpp.

Referenced by [drawRubberBand\(\)](#), and [QwtPlotPicker::end\(\)](#).

6.41.4.3 void QwtPicker::setRubberBand ([RubberBand](#) *rubberBand*) [virtual]

Set the rubberband style

Parameters:

rubberBand Rubberband style The default value is NoRubberBand.

See also:

[rubberBand\(\)](#), [RubberBand](#), [setRubberBandPen\(\)](#)

Definition at line 427 of file qwt_picker.cpp.

6.41.4.4 [QwtPicker::RubberBand](#) QwtPicker::rubberBand () const

Returns:

Rubberband style

See also:

[setRubberBand\(\)](#), [RubberBand](#), [rubberBandPen\(\)](#)

Definition at line 436 of file `qwt_picker.cpp`.

Referenced by `drawRubberBand()`, `trackerRect()`, `QwtPlotPicker::trackerText()`, `trackerText()`, and `updateDisplay()`.

6.41.4.5 void QwtPicker::setTrackerMode ([DisplayMode mode](#)) [virtual]

Set the display mode of the tracker.

A tracker displays information about current position of the cursor as a string. The display mode controls if the tracker has to be displayed whenever the observed widget has focus and cursor (`AlwaysOn`), never (`AlwaysOff`), or only when the selection is active (`ActiveOnly`).

Parameters:

mode Tracker display mode

Warning:

In case of `AlwaysOn`, `mouseTracking` will be enabled for the observed widget.

See also:

[trackerMode\(\)](#), [DisplayMode](#)

Definition at line 457 of file `qwt_picker.cpp`.

6.41.4.6 [QwtPicker::DisplayMode QwtPicker::trackerMode \(\)](#) const

Returns:

Tracker display mode

See also:

[setTrackerMode\(\)](#), [DisplayMode](#)

Definition at line 470 of file `qwt_picker.cpp`.

Referenced by `begin()`, `end()`, `trackerRect()`, and `updateDisplay()`.

6.41.4.7 void QwtPicker::setResizeMode ([ResizeMode mode](#)) [virtual]

Set the resize mode.

The resize mode controls what to do with the selected points of an active selection when the observed widget is resized.

Stretch means the points are scaled according to the new size, `KeepSize` means the points remain unchanged.

The default mode is `Stretch`.

Parameters:

mode Resize mode

See also:

[resizeMode\(\)](#), [SizeMode](#)

Definition at line 489 of file qwt_picker.cpp.

6.41.4.8 [QwtPicker::SizeMode](#) QwtPicker::resizeMode () const

Returns:

Resize mode

See also:

[setSizeMode\(\)](#), [SizeMode](#)

Definition at line 499 of file qwt_picker.cpp.

6.41.4.9 void QwtPicker::setRubberBandPen (const QPen & *pen*) [virtual]

Set the pen for the rubberband

Parameters:

pen Rubberband pen

See also:

[rubberBandPen\(\)](#), [setRubberBand\(\)](#)

Definition at line 597 of file qwt_picker.cpp.

References [updateDisplay\(\)](#).

6.41.4.10 QPen QwtPicker::rubberBandPen () const

Returns:

Rubberband pen

See also:

[setRubberBandPen\(\)](#), [rubberBand\(\)](#)

Definition at line 610 of file qwt_picker.cpp.

Referenced by [drawRubberBand\(\)](#), and [updateDisplay\(\)](#).

6.41.4.11 void QwtPicker::setTrackerPen (const QPen & *pen*) [virtual]

Set the pen for the tracker

Parameters:

pen Tracker pen

See also:

[trackerPen\(\)](#), [setTrackerMode\(\)](#), [setTrackerFont\(\)](#)

Definition at line 573 of file qwt_picker.cpp.

References [updateDisplay\(\)](#).

6.41.4.12 QPen QwtPicker::trackerPen () const

Returns:

Tracker pen

See also:

[setTrackerPen\(\)](#), [trackerMode\(\)](#), [trackerFont\(\)](#)

Definition at line 586 of file qwt_picker.cpp.

Referenced by [updateDisplay\(\)](#).

6.41.4.13 void QwtPicker::setTrackerFont (const QFont & *font*) [virtual]

Set the font for the tracker

Parameters:

font Tracker font

See also:

[trackerFont\(\)](#), [setTrackerMode\(\)](#), [setTrackerPen\(\)](#)

Definition at line 548 of file qwt_picker.cpp.

References [updateDisplay\(\)](#).

6.41.4.14 QFont QwtPicker::trackerFont () const

Returns:

Tracker font

See also:

[setTrackerFont\(\)](#), [trackerMode\(\)](#), [trackerPen\(\)](#)

Definition at line 562 of file qwt_picker.cpp.

6.41.4.15 bool QwtPicker::isEnabled () const

Returns:

true when enabled, false otherwise

See also:

[setEnabled\(\)](#), [eventFilter\(\)](#)

Definition at line 537 of file qwt_picker.cpp.

6.41.4.16 void QwtPicker::setEnabled (bool *enabled*) [virtual]

En/disable the picker.

When enabled is true an event filter is installed for the observed widget, otherwise the event filter is removed.

Parameters:

enabled true or false

See also:

[isEnabled\(\)](#), [eventFilter\(\)](#)

Definition at line 513 of file qwt_picker.cpp.

References [parentWidget\(\)](#), and [updateDisplay\(\)](#).

6.41.4.17 bool QwtPicker::isActive () const

A picker is active between [begin\(\)](#) and [end\(\)](#).

Returns:

true if the selection is active.

Definition at line 1265 of file qwt_picker.cpp.

Referenced by [drawRubberBand\(\)](#), [reset\(\)](#), [trackerRect\(\)](#), [updateDisplay\(\)](#), [QwtPlotZoomer::widgetKeyPressEvent\(\)](#), [widgetLeaveEvent\(\)](#), and [widgetMouseMoveEvent\(\)](#).

6.41.4.18 bool QwtPicker::eventFilter (QObject * *o*, QEvent * *e*) [virtual]

Event filter.

When [isEnabled\(\)](#) == true all events of the observed widget are filtered. Mouse and keyboard events are translated into [widgetMouse-](#) and [widgetKey-](#) and [widgetWheel-](#)events. Paint and Resize events are handled to keep rubberband and tracker up to date.

See also:

[event\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 852 of file qwt_picker.cpp.

References [parentWidget\(\)](#), [stretchSelection\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#), [widgetLeaveEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), and [widgetWheelEvent\(\)](#).

6.41.4.19 QWidget * QwtPicker::parentWidget ()

Return the parent widget, where the selection happens.

Definition at line 375 of file qwt_picker.cpp.

Referenced by [begin\(\)](#), [QwtPlotPicker::canvas\(\)](#), [eventFilter\(\)](#), [pickRect\(\)](#), [setEnabled\(\)](#), [transition\(\)](#), [updateDisplay\(\)](#), and [widgetKeyPressEvent\(\)](#).

6.41.4.20 const QWidget * QwtPicker::parentWidget () const

Return the parent widget, where the selection happens.

Definition at line 385 of file qwt_picker.cpp.

6.41.4.21 QRect QwtPicker::pickRect () const [virtual]

Find the area of the observed widget, where selection might happen.

Returns:

QFrame::contentsRect() if it is a QFrame, QWidget::rect() otherwise.

Definition at line 1344 of file qwt_picker.cpp.

References parentWidget().

Referenced by drawRubberBand(), trackerRect(), widgetKeyPressEvent(), widgetMouseMoveEvent(), and widgetWheelEvent().

6.41.4.22 const QwtPolygon & QwtPicker::selection () const

Return Selected points.

Definition at line 1271 of file qwt_picker.cpp.

Referenced by QwtPlotZoomer::end(), and QwtPlotPicker::end().

6.41.4.23 void QwtPicker::drawRubberBand (QPainter * *painter*) const [virtual]

Draw a rubberband , depending on [rubberBand\(\)](#) and [selectionFlags\(\)](#)

Parameters:

painter Painter, initialized with clip rect

See also:

[rubberBand\(\)](#), [RubberBand](#), [selectionFlags\(\)](#)

Definition at line 654 of file qwt_picker.cpp.

References QPainter::drawEllipse(), QPainter::drawLine(), QPainter::drawRect(), isActive(), pickRect(), rubberBand(), rubberBandPen(), and selectionFlags().

6.41.4.24 void QwtPicker::drawTracker (QPainter * *painter*) const [virtual]

Draw the tracker

Parameters:

painter Painter

See also:

[trackerRect\(\)](#), [trackerText\(\)](#)

Definition at line 749 of file qwt_picker.cpp.

References QwtText::draw(), QwtText::isEmpty(), QwtText::setFont(), trackerRect(), trackerText(), and QwtText::usedFont().

6.41.4.25 [QwtText](#) QwtPicker::trackerText (const QPoint & *pos*) const [virtual]

Return the label for a position.

In case of HLineRubberBand the label is the value of the y position, in case of VLineRubberBand the value of the x position. Otherwise the label contains x and y position separated by a ','.

The format for the string conversion is "%d".

Parameters:

pos Position

Returns:

Converted position as string

Reimplemented in [QwtPlotPicker](#).

Definition at line 628 of file qwt_picker.cpp.

References [rubberBand\(\)](#).

Referenced by [drawTracker\(\)](#), and [trackerRect\(\)](#).

6.41.4.26 void QwtPicker::selected (const QwtPolygon & *pa*) [signal]

A signal emitting the selected points, at the end of a selection.

Parameters:

pa Selected points

Referenced by [end\(\)](#).

6.41.4.27 void QwtPicker::appended (const QPoint & *pos*) [signal]

A signal emitted when a point has been appended to the selection

Parameters:

pos Position of the appended point.

See also:

[append\(\)](#), [moved\(\)](#)

Referenced by [append\(\)](#).

6.41.4.28 void QwtPicker::moved (const QPoint & *pos*) [signal]

A signal emitted whenever the last appended point of the selection has been moved.

Parameters:

pos Position of the moved last point of the selection.

See also:

[move\(\)](#), [appended\(\)](#)

Referenced by [move\(\)](#).

6.41.4.29 void QwtPicker::changed (const QwtPolygon & *pa*) [signal]

A signal emitted when the active selection has been changed. This might happen when the observed widget is resized.

Parameters:

pa Changed selection

See also:

[stretchSelection\(\)](#)

Referenced by [stretchSelection\(\)](#).

6.41.4.30 bool QwtPicker::accept (QwtPolygon & *selection*) const [protected, virtual]

Validate and fixup the selection.

Accepts all selections unmodified

Parameters:

selection Selection to validate and fixup

Returns:

true, when accepted, false otherwise

Reimplemented in [QwtPlotZoomer](#).

Definition at line 1256 of file `qwt_picker.cpp`.

Referenced by [end\(\)](#).

6.41.4.31 void QwtPicker::transition (const QEvent * *e*) [protected, virtual]

Passes an event to the state machine and executes the resulting commands. Append and Move commands use the current position of the cursor (`QCursor::pos()`).

Parameters:

e Event

Definition at line 1079 of file `qwt_picker.cpp`.

References [append\(\)](#), [begin\(\)](#), [end\(\)](#), [move\(\)](#), and [parentWidget\(\)](#).

Referenced by [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), and [widgetWheelEvent\(\)](#).

6.41.4.32 void QwtPicker::begin () [protected, virtual]

Open a selection setting the state to active

See also:

[isActive](#), [end\(\)](#), [append\(\)](#), [move\(\)](#)

Reimplemented in [QwtPlotZoomer](#).

Definition at line 1136 of file `qwt_picker.cpp`.

References `parentWidget()`, `trackerMode()`, and `updateDisplay()`.

Referenced by `QwtPlotZoomer::begin()`, and `transition()`.

6.41.4.33 void QwtPicker::append (const QPoint & pos) [protected, virtual]

Append a point to the selection and update rubberband and tracker. The [appended\(\)](#) signal is emitted.

Parameters:

pos Additional point

See also:

[isActive](#), [begin\(\)](#), [end\(\)](#), [move\(\)](#), [appended\(\)](#)

Reimplemented in [QwtPlotPicker](#).

Definition at line 1215 of file `qwt_picker.cpp`.

References `appended()`, and `updateDisplay()`.

Referenced by `QwtPlotPicker::append()`, and `transition()`.

6.41.4.34 void QwtPicker::move (const QPoint & pos) [protected, virtual]

Move the last point of the selection The [moved\(\)](#) signal is emitted.

Parameters:

pos New position

See also:

[isActive](#), [begin\(\)](#), [end\(\)](#), [append\(\)](#)

Reimplemented in [QwtPlotPicker](#).

Definition at line 1237 of file `qwt_picker.cpp`.

References `moved()`, and `updateDisplay()`.

Referenced by `QwtPlotPicker::move()`, and `transition()`.

6.41.4.35 bool QwtPicker::end (bool ok = true) [protected, virtual]

Close a selection setting the state to inactive.

The selection is validated and maybe fixed by [QwtPicker::accept\(\)](#).

Parameters:

ok If true, complete the selection and emit a selected signal otherwise discard the selection.

Returns:

true if the selection is accepted, false otherwise

See also:

[isActive](#), [begin\(\)](#), [append\(\)](#), [move\(\)](#), [selected\(\)](#), [accept\(\)](#)

Reimplemented in [QwtPlotPicker](#), and [QwtPlotZoomer](#).

Definition at line 1168 of file `qwt_picker.cpp`.

References [accept\(\)](#), [selected\(\)](#), [trackerMode\(\)](#), and [updateDisplay\(\)](#).

Referenced by [QwtPlotPicker::end\(\)](#), [reset\(\)](#), and [transition\(\)](#).

6.41.4.36 void QwtPicker::reset () [protected, virtual]

Reset the state machine and terminate (`end(false)`) the selection

Definition at line 1198 of file `qwt_picker.cpp`.

References [end\(\)](#), and [isActive\(\)](#).

Referenced by [widgetKeyPressEvent\(\)](#).

6.41.4.37 void QwtPicker::widgetMouseEvent (QMouseEvent * e) [protected, virtual]

Handle a mouse press event for the observed widget.

Begin and/or end a selection depending on the selection flags.

See also:

[QwtPicker](#), [selectionFlags\(\)](#),
[eventFilter\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 912 of file `qwt_picker.cpp`.

References [transition\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.38 void QwtPicker::widgetMouseReleaseEvent (QMouseEvent * e) [protected, virtual]

Handle a mouse release event for the observed widget.

End a selection depending on the selection flags.

See also:

[QwtPicker](#), [selectionFlags\(\)](#),
[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Reimplemented in [QwtPlotZoomer](#).

Definition at line 963 of file `qwt_picker.cpp`.

References [transition\(\)](#).

Referenced by [eventFilter\(\)](#), and [QwtPlotZoomer::widgetMouseReleaseEvent\(\)](#).

6.41.4.39 void QwtPicker::widgetMouseDoubleClickEvent (QMouseEvent * *me*) [protected, virtual]

Handle mouse double click event for the observed widget.

Empty implementation, does nothing.

See also:

[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 977 of file qwt_picker.cpp.

References [transition\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.40 void QwtPicker::widgetMouseMoveEvent (QMouseEvent * *e*) [protected, virtual]

Handle a mouse move event for the observed widget.

Move the last point of the selection in case of [isActive\(\)](#) == true

See also:

[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 926 of file qwt_picker.cpp.

References [isActive\(\)](#), [pickRect\(\)](#), [transition\(\)](#), and [updateDisplay\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.41 void QwtPicker::widgetWheelEvent (QWheelEvent * *e*) [protected, virtual]

Handle a wheel event for the observed widget.

Move the last point of the selection in case of [isActive\(\)](#) == true

See also:

[eventFilter\(\)](#), [widgetMousePressEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 992 of file qwt_picker.cpp.

References [pickRect\(\)](#), [transition\(\)](#), and [updateDisplay\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.42 void QwtPicker::widgetKeyPressEvent (QKeyEvent * *ke*) [protected, virtual]

Handle a key press event for the observed widget.

Selections can be completely done by the keyboard. The arrow keys move the cursor, the abort key aborts a selection. All other keys are handled by the current state machine.

See also:

[QwtPicker](#), [selectionFlags\(\)](#)
[eventFilter\(\)](#), [widgetMouseEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#), [stateMachine\(\)](#),
[QwtEventPattern::KeyPatternCode](#)

Reimplemented in [QwtPlotZoomer](#).

Definition at line 1017 of file `qwt_picker.cpp`.

References [QwtEventPattern::keyMatch\(\)](#), [parentWidget\(\)](#), [pickRect\(\)](#), [reset\(\)](#), and [transition\(\)](#).

Referenced by [eventFilter\(\)](#), and [QwtPlotZoomer::widgetKeyPressEvent\(\)](#).

6.41.4.43 `void QwtPicker::widgetKeyReleaseEvent (QKeyEvent * ke) [protected, virtual]`

Handle a key release event for the observed widget.

Passes the event to the state machine.

See also:

[eventFilter\(\)](#), [widgetMouseEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetMouseMoveEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [stateMachine\(\)](#)

Definition at line 1067 of file `qwt_picker.cpp`.

References [transition\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.44 `void QwtPicker::widgetLeaveEvent (QEvent *) [protected, virtual]`

Handle a leave event for the observed widget.

See also:

[eventFilter\(\)](#), [widgetMouseEvent\(\)](#), [widgetMouseReleaseEvent\(\)](#), [widgetMouseDoubleClickEvent\(\)](#), [widgetWheelEvent\(\)](#), [widgetKeyPressEvent\(\)](#), [widgetKeyReleaseEvent\(\)](#)

Definition at line 946 of file `qwt_picker.cpp`.

References [isActive\(\)](#), and [updateDisplay\(\)](#).

Referenced by [eventFilter\(\)](#).

6.41.4.45 `void QwtPicker::stretchSelection (const QSize & oldSize, const QSize & newSize) [protected, virtual]`

Scale the selection by the ratios of `oldSize` and `newSize` The [changed\(\)](#) signal is emitted.

Parameters:

oldSize Previous size

newSize Current size

See also:

[ResizeMode](#), [setSizeMode\(\)](#), [resizeMode\(\)](#)

Definition at line 1285 of file qwt_picker.cpp.

References changed().

Referenced by eventFilter().

6.41.4.46 QwtPickerMachine * QwtPicker::stateMachine (int flags) const [protected, virtual]

Create a state machine depending on the selection flags.

- PointSelection | ClickSelection
QwtPickerClickPointMachine()
- PointSelection | DragSelection
QwtPickerDragPointMachine()
- RectSelection | ClickSelection
QwtPickerClickRectMachine()
- RectSelection | DragSelection
QwtPickerDragRectMachine()
- PolygonSelection
QwtPickerPolygonMachine()

See also:

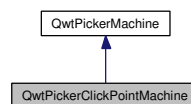
[setSelectionFlags\(\)](#)

Definition at line 351 of file qwt_picker.cpp.

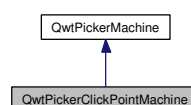
Referenced by setSelectionFlags().

6.42 QwtPickerClickPointMachine Class Reference

Inheritance diagram for QwtPickerClickPointMachine:



Collaboration diagram for QwtPickerClickPointMachine:



6.42.1 Detailed Description

A state machine for point selections.

Pressing `QwtEventPattern::MouseSelect1` or `QwtEventPattern::KeySelect1` selects a point.

See also:

[QwtEventPattern::MousePatternCode](#), [QwtEventPattern::KeyPatternCode](#)

Definition at line 75 of file `qwt_picker_machine.h`.

Public Member Functions

- virtual [CommandList transition](#) (const [QwtEventPattern](#) &, const `QEvent *`)

6.42.2 Member Function Documentation

6.42.2.1 [QwtPickerMachine::CommandList](#) [QwtPickerClickPointMachine::transition](#) (const [QwtEventPattern](#) &, const `QEvent *`) `[virtual]`

Transition.

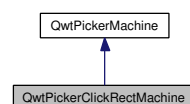
Implements [QwtPickerMachine](#).

Definition at line 44 of file `qwt_picker_machine.cpp`.

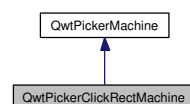
References [QwtEventPattern::keyMatch\(\)](#), and [QwtEventPattern::mouseMatch\(\)](#).

6.43 QwtPickerClickRectMachine Class Reference

Inheritance diagram for `QwtPickerClickRectMachine`:



Collaboration diagram for `QwtPickerClickRectMachine`:



6.43.1 Detailed Description

A state machine for rectangle selections.

Pressing `QwtEventPattern::MouseSelect1` starts the selection, releasing it selects the first point. Pressing it again selects the second point and terminates the selection. Pressing `QwtEventPattern::KeySelect1` also starts the selection, a second press selects the first point. A third one selects the second point and terminates the selection.

See also:

[QwtEventPattern::MousePatternCode](#), [QwtEventPattern::KeyPatternCode](#)

Definition at line 109 of file qwt_picker_machine.h.

Public Member Functions

- virtual [CommandList transition](#) (const [QwtEventPattern](#) &, const [QEvent](#) *)

6.43.2 Member Function Documentation

6.43.2.1 [QwtPickerMachine::CommandList](#) [QwtPickerClickRectMachine::transition](#) (const [QwtEventPattern](#) &, const [QEvent](#) *) [virtual]

Transition.

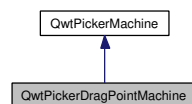
Implements [QwtPickerMachine](#).

Definition at line 145 of file qwt_picker_machine.cpp.

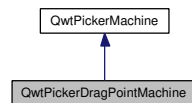
References [QwtEventPattern::keyMatch\(\)](#), [QwtEventPattern::mouseMatch\(\)](#), [QwtPickerMachine::setState\(\)](#), and [QwtPickerMachine::state\(\)](#).

6.44 QwtPickerDragPointMachine Class Reference

Inheritance diagram for QwtPickerDragPointMachine:



Collaboration diagram for QwtPickerDragPointMachine:



6.44.1 Detailed Description

A state machine for point selections.

Pressing [QwtEventPattern::MouseSelect1](#) or [QwtEventPattern::KeySelect1](#) starts the selection, releasing [QwtEventPattern::MouseSelect1](#) or a second press of [QwtEventPattern::KeySelect1](#) terminates it.

Definition at line 89 of file qwt_picker_machine.h.

Public Member Functions

- virtual [CommandList transition](#) (const [QwtEventPattern](#) &, const [QEvent](#) *)

6.44.2 Member Function Documentation

6.44.2.1 [QwtPickerMachine::CommandList](#) QwtPickerDragPointMachine::transition (const [QwtEventPattern](#) &, const [QEvent](#) *) [virtual]

Transition.

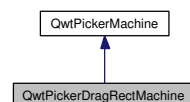
Implements [QwtPickerMachine](#).

Definition at line 81 of file qwt_picker_machine.cpp.

References [QwtEventPattern::keyMatch\(\)](#), [QwtEventPattern::mouseMatch\(\)](#), [QwtPickerMachine::setState\(\)](#), and [QwtPickerMachine::state\(\)](#).

6.45 QwtPickerDragRectMachine Class Reference

Inheritance diagram for QwtPickerDragRectMachine:



Collaboration diagram for QwtPickerDragRectMachine:



6.45.1 Detailed Description

A state machine for rectangle selections.

Pressing `QwtEventPattern::MouseSelect1` selects the first point, releasing it the second point. Pressing `QwtEventPattern::KeySelect1` also selects the first point, a second press selects the second point and terminates the selection.

See also:

[QwtEventPattern::MousePatternCode](#), [QwtEventPattern::KeyPatternCode](#)

Definition at line 128 of file qwt_picker_machine.h.

Public Member Functions

- virtual [CommandList](#) transition (const [QwtEventPattern](#) &, const [QEvent](#) *)

6.45.2 Member Function Documentation

6.45.2.1 [QwtPickerMachine::CommandList](#) QwtPickerDragRectMachine::transition (const [QwtEventPattern](#) &, const [QEvent](#) *) [virtual]

Transition.

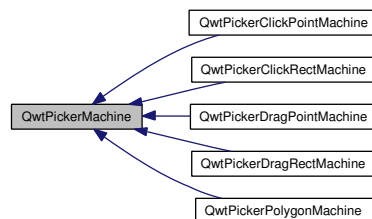
Implements [QwtPickerMachine](#).

Definition at line 234 of file qwt_picker_machine.cpp.

References [QwtEventPattern::keyMatch\(\)](#), [QwtEventPattern::mouseMatch\(\)](#), [QwtPickerMachine::setState\(\)](#), and [QwtPickerMachine::state\(\)](#).

6.46 QwtPickerMachine Class Reference

Inheritance diagram for QwtPickerMachine:



6.46.1 Detailed Description

A state machine for [QwtPicker](#) selections.

[QwtPickerMachine](#) accepts key and mouse events and translates them into selection commands.

See also:

[QwtEventPattern::MousePatternCode](#), [QwtEventPattern::KeyPatternCode](#)

Definition at line 32 of file qwt_picker_machine.h.

Public Types

- enum [Command](#) {
 Begin,
 Append,
 Move,
 End }
- typedef `QList< Command >` [CommandList](#)

Public Member Functions

- virtual `~QwtPickerMachine ()`
- virtual `CommandList transition (const QwtEventPattern &, const QEvent *)=0`
- void `reset ()`
- int `state () const`
- void `setState (int)`

Protected Member Functions

- [QwtPickerMachine](#) ()

6.46.2 Member Enumeration Documentation

6.46.2.1 enum [QwtPickerMachine::Command](#)

Commands - the output of the state machine.

Definition at line 36 of file `qwt_picker_machine.h`.

6.46.3 Constructor & Destructor Documentation

6.46.3.1 [QwtPickerMachine::~~QwtPickerMachine](#) () [virtual]

Destructor.

Definition at line 21 of file `qwt_picker_machine.cpp`.

6.46.3.2 [QwtPickerMachine::QwtPickerMachine](#) () [protected]

Constructor.

Definition at line 15 of file `qwt_picker_machine.cpp`.

6.46.4 Member Function Documentation

6.46.4.1 virtual [CommandList](#) [QwtPickerMachine::transition](#) (const [QwtEventPattern](#) &, const [QEvent](#) *) [pure virtual]

Transition.

Implemented in [QwtPickerClickPointMachine](#), [QwtPickerDragPointMachine](#), [QwtPickerClickRectMachine](#), [QwtPickerDragRectMachine](#), and [QwtPickerPolygonMachine](#).

6.46.4.2 void [QwtPickerMachine::reset](#) ()

Set the current state to 0.

Definition at line 38 of file `qwt_picker_machine.cpp`.

References `setState()`.

6.46.4.3 int [QwtPickerMachine::state](#) () const

Return the current state.

Definition at line 26 of file `qwt_picker_machine.cpp`.

Referenced by `QwtPickerPolygonMachine::transition()`, `QwtPickerDragRectMachine::transition()`, `QwtPickerClickRectMachine::transition()`, and `QwtPickerDragPointMachine::transition()`.

6.46.4.4 void [QwtPickerMachine::setState](#) (int)

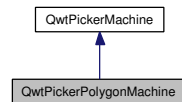
Change the current state.

Definition at line 32 of file qwt_picker_machine.cpp.

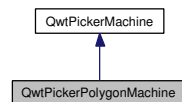
Referenced by `reset()`, `QwtPickerPolygonMachine::transition()`, `QwtPickerDragRectMachine::transition()`, `QwtPickerClickRectMachine::transition()`, and `QwtPickerDragPointMachine::transition()`.

6.47 QwtPickerPolygonMachine Class Reference

Inheritance diagram for QwtPickerPolygonMachine:



Collaboration diagram for QwtPickerPolygonMachine:



6.47.1 Detailed Description

A state machine for polygon selections.

Pressing `QwtEventPattern::MouseSelect1` or `QwtEventPattern::KeySelect1` starts the selection and selects the first point, or appends a point. Pressing `QwtEventPattern::MouseSelect2` or `QwtEventPattern::KeySelect2` appends the last point and terminates the selection.

See also:

[QwtEventPattern::MousePatternCode](#), [QwtEventPattern::KeyPatternCode](#)

Definition at line 146 of file qwt_picker_machine.h.

Public Member Functions

- virtual [CommandList transition](#) (const [QwtEventPattern](#) &, const `QEvent *`)

6.47.2 Member Function Documentation

6.47.2.1 [QwtPickerMachine::CommandList](#) `QwtPickerPolygonMachine::transition` (const [QwtEventPattern](#) &, const `QEvent *`) `[virtual]`

Transition.

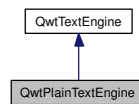
Implements [QwtPickerMachine](#).

Definition at line 300 of file qwt_picker_machine.cpp.

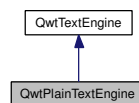
References `QwtEventPattern::keyMatch()`, `QwtEventPattern::mouseMatch()`, `QwtPickerMachine::setState()`, and `QwtPickerMachine::state()`.

6.48 QwtPlainTextEngine Class Reference

Inheritance diagram for QwtPlainTextEngine:



Collaboration diagram for QwtPlainTextEngine:



6.48.1 Detailed Description

A text engine for plain texts.

[QwtPlainTextEngine](#) renders texts using the basic Qt classes QPainter and QFontMetrics.

Definition at line 116 of file qwt_text_engine.h.

Public Member Functions

- [QwtPlainTextEngine](#) ()
- virtual [~QwtPlainTextEngine](#) ()
- virtual int [heightForWidth](#) (const QFont &font, int flags, const QString &text, int width) const
- virtual QSize [textSize](#) (const QFont &font, int flags, const QString &text) const
- virtual void [draw](#) (QPainter *painter, const QRect &rect, int flags, const QString &text) const
- virtual bool [mightRender](#) (const QString &) const
- virtual void [textMargins](#) (const QFont &, const QString &, int &left, int &right, int &top, int &bottom) const

6.48.2 Constructor & Destructor Documentation

6.48.2.1 QwtPlainTextEngine::QwtPlainTextEngine ()

Constructor.

Definition at line 180 of file qwt_text_engine.cpp.

6.48.2.2 QwtPlainTextEngine::~~QwtPlainTextEngine () [virtual]

Destructor.

Definition at line 186 of file qwt_text_engine.cpp.

6.48.3 Member Function Documentation

6.48.3.1 `int QwtPlainTextEngine::heightForWidth (const QFont & font, int flags, const QString & text, int width) const` [virtual]

Find the height for a given width

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

width Width

Returns:

Calculated height

Implements [QwtTextEngine](#).

Definition at line 201 of file qwt_text_engine.cpp.

6.48.3.2 `QSize QwtPlainTextEngine::textSize (const QFont & font, int flags, const QString & text) const` [virtual]

Returns the size, that is needed to render text

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

Returns:

Calculated size

Implements [QwtTextEngine](#).

Definition at line 220 of file qwt_text_engine.cpp.

6.48.3.3 `void QwtPlainTextEngine::draw (QPainter * painter, const QRect & rect, int flags, const QString & text) const` [virtual]

Draw the text in a clipping rectangle.

A wrapper for QPainter::drawText.

Parameters:

painter Painter

rect Clipping rectangle

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

Implements [QwtTextEngine](#).

Definition at line 259 of file qwt_text_engine.cpp.

References [QwtPainter::drawText\(\)](#).

6.48.3.4 bool QwtPlainTextEngine::mightRender (const QString &) const [virtual]

Test if a string can be rendered by this text engine.

Returns:

Always true. All texts can be rendered by [QwtPlainTextEngine](#)

Implements [QwtTextEngine](#).

Definition at line 269 of file qwt_text_engine.cpp.

6.48.3.5 void QwtPlainTextEngine::textMargins (const QFont & font, const QString &, int & left, int & right, int & top, int & bottom) const [virtual]

Return margins around the texts

Parameters:

font Font of the text

left Return 0

right Return 0

top Return value for the top margin

bottom Return value for the bottom margin

Implements [QwtTextEngine](#).

Definition at line 239 of file qwt_text_engine.cpp.

6.49 QwtPlot Class Reference

Inheritance diagram for QwtPlot:



Collaboration diagram for QwtPlot:



6.49.1 Detailed Description

A 2-D plotting widget.

[QwtPlot](#) is a widget for plotting two-dimensional graphs. An unlimited number of plot items can be displayed on its canvas. Plot items might be curves ([QwtPlotCurve](#)), markers ([QwtPlotMarker](#)), the grid ([QwtPlotGrid](#)), or anything else derived from [QwtPlotItem](#). A plot can have up to four axes, with each

plot item attached to an x- and a y axis. The scales at the axes can be explicitly set ([QwtScaleDiv](#)), or are calculated from the plot items, using algorithms ([QwtScaleEngine](#)) which can be configured separately for each axis.

Example

The following example shows (schematically) the most simple way to use [QwtPlot](#). By default, only the left and bottom axes are visible and their scales are computed automatically.

```
#include <qwt_plot.h>
#include <qwt_plot_curve.h>

QwtPlot *myPlot;
double x[100], y1[100], y2[100];          // x and y values

myPlot = new QwtPlot("Two Curves", parent);

// add curves
QwtPlotCurve *curve1 = new QwtPlotCurve("Curve 1");
QwtPlotCurve *curve2 = new QwtPlotCurve("Curve 2");

getSomeValues(x, y1, y2);

// copy the data into the curves
curve1->setData(x, y1, 100);
curve2->setData(x, y2, 100);

curve1->attach(myPlot);
curve2->attach(myPlot);

// finally, refresh the plot
myPlot->replot();
```

Definition at line 77 of file `qwt_plot.h`.

Public Types

- enum [Axis](#) {
 yLeft,
 yRight,
 xBottom,
 xTop,
 axisCnt }
- enum [LegendPosition](#) {
 LeftLegend,
 RightLegend,
 BottomLegend,
 TopLegend,
 ExternalLegend }

Public Slots

- virtual void [clear](#) ()
- virtual void [replot](#) ()
- void [autoRefresh](#) ()

Signals

- void [legendClicked](#) ([QwtPlotItem](#) *plotItem)
- void [legendChecked](#) ([QwtPlotItem](#) *plotItem, bool on)

Public Member Functions

- [QwtPlot](#) ([QWidget](#) *p=NULL)
- [QwtPlot](#) (const [QwtText](#) &title, [QWidget](#) *p=NULL)
- virtual [~QwtPlot](#) ()
- void [applyProperties](#) (const [QString](#) &)
- [QString](#) [grabProperties](#) () const
- void [setAutoReplot](#) (bool tf=true)
- bool [autoReplot](#) () const
- void [print](#) ([QPaintDevice](#) &p, const [QwtPlotPrintFilter](#) &=QwtPlotPrintFilter()) const
- virtual void [print](#) ([QPainter](#) *, const [QRect](#) &rect, const [QwtPlotPrintFilter](#) &=QwtPlotPrintFilter()) const
- [QwtPlotLayout](#) * [plotLayout](#) ()
- const [QwtPlotLayout](#) * [plotLayout](#) () const
- void [setMargin](#) (int margin)
- int [margin](#) () const
- void [setTitle](#) (const [QString](#) &)
- void [setTitle](#) (const [QwtText](#) &t)
- [QwtText](#) [title](#) () const
- [QwtTextLabel](#) * [titleLabel](#) ()
- const [QwtTextLabel](#) * [titleLabel](#) () const
- [QwtPlotCanvas](#) * [canvas](#) ()
- const [QwtPlotCanvas](#) * [canvas](#) () const
- void [setCanvasBackground](#) (const [QColor](#) &c)
- const [QColor](#) & [canvasBackground](#) () const
- void [setCanvasLineWidth](#) (int w)
- int [canvasLineWidth](#) () const
- virtual [QwtScaleMap](#) [canvasMap](#) (int axisId) const
- double [invTransform](#) (int axisId, int pos) const
- int [transform](#) (int axisId, double value) const
- [QwtScaleEngine](#) * [axisScaleEngine](#) (int axisId)
- const [QwtScaleEngine](#) * [axisScaleEngine](#) (int axisId) const
- void [setAxisScaleEngine](#) (int axisId, [QwtScaleEngine](#) *)
- void [setAxisAutoScale](#) (int axisId)
- bool [axisAutoScale](#) (int axisId) const
- void [enableAxis](#) (int axisId, bool tf=true)
- bool [axisEnabled](#) (int axisId) const
- void [setAxisFont](#) (int axisId, const [QFont](#) &f)
- [QFont](#) [axisFont](#) (int axisId) const
- void [setAxisScale](#) (int axisId, double min, double max, double step=0)
- void [setAxisScaleDiv](#) (int axisId, const [QwtScaleDiv](#) &)
- void [setAxisScaleDraw](#) (int axisId, [QwtScaleDraw](#) *)
- double [axisStepSize](#) (int axisId) const
- const [QwtScaleDiv](#) * [axisScaleDiv](#) (int axisId) const
- [QwtScaleDiv](#) * [axisScaleDiv](#) (int axisId)

- const [QwtScaleDraw](#) * [axisScaleDraw](#) (int axisId) const
- [QwtScaleDraw](#) * [axisScaleDraw](#) (int axisId)
- const [QwtScaleWidget](#) * [axisWidget](#) (int axisId) const
- [QwtScaleWidget](#) * [axisWidget](#) (int axisId)
- void [setAxisLabelAlignment](#) (int axisId, Qt::Alignment)
- void [setAxisLabelRotation](#) (int axisId, double rotation)
- void [setAxisTitle](#) (int axisId, const QString &)
- void [setAxisTitle](#) (int axisId, const [QwtText](#) &)
- [QwtText](#) [axisTitle](#) (int axisId) const
- void [setAxisMaxMinor](#) (int axisId, int maxMinor)
- int [axisMaxMajor](#) (int axisId) const
- void [setAxisMaxMajor](#) (int axisId, int maxMajor)
- int [axisMaxMinor](#) (int axisId) const
- void [insertLegend](#) ([QwtLegend](#) *, [LegendPosition](#)=QwtPlot::RightLegend, double ratio=-1.0)
- [QwtLegend](#) * [legend](#) ()
- const [QwtLegend](#) * [legend](#) () const
- virtual void [polish](#) ()
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- virtual void [updateLayout](#) ()
- virtual bool [event](#) (QEvent *)

Protected Slots

- virtual void [legendItemClicked](#) ()
- virtual void [legendItemChecked](#) (bool)

Protected Member Functions

- virtual void [drawCanvas](#) (QPainter *)
- virtual void [drawItems](#) (QPainter *, const QRect &, const [QwtScaleMap](#) maps[axisCnt], const [QwtPlotPrintFilter](#) &) const
- virtual void [updateTabOrder](#) ()
- void [updateAxes](#) ()
- virtual void [resizeEvent](#) (QResizeEvent *e)
- virtual void [printLegendItem](#) (QPainter *, const QWidget *, const QRect &) const
- virtual void [printTitle](#) (QPainter *, const QRect &) const
- virtual void [printScale](#) (QPainter *, int axisId, int startDist, int endDist, int baseDist, const QRect &) const
- virtual void [printCanvas](#) (QPainter *, const QRect &boundingRect, const QRect &canvasRect, const [QwtScaleMap](#) maps[axisCnt], const [QwtPlotPrintFilter](#) &) const
- virtual void [printLegend](#) (QPainter *, const QRect &) const

Static Protected Member Functions

- static bool [axisValid](#) (int axisId)

Friends

- class [QwtPlotCanvas](#)

6.49.2 Member Enumeration Documentation

6.49.2.1 enum [QwtPlot::Axis](#)

Axis index.

Definition at line 87 of file qwt_plot.h.

6.49.2.2 enum [QwtPlot::LegendPosition](#)

Position of the legend, relative to the canvas.

ExternalLegend means that only the content of the legend will be handled by [QwtPlot](#), but not its geometry. This might be interesting if an application wants to have a legend in an external window.

Definition at line 105 of file qwt_plot.h.

6.49.3 Constructor & Destructor Documentation

6.49.3.1 [QwtPlot::QwtPlot \(QWidget *parent = NULL\)](#) [explicit]

Constructor.

Parameters:

parent Parent widget

Definition at line 52 of file qwt_plot.cpp.

6.49.3.2 [QwtPlot::QwtPlot \(const \[QwtText\]\(#\) &title, QWidget *parent = NULL\)](#) [explicit]

Constructor.

Parameters:

title Title text

parent Parent widget

Definition at line 63 of file qwt_plot.cpp.

References [title\(\)](#).

6.49.3.3 [QwtPlot::~~QwtPlot \(\)](#) [virtual]

Destructor.

Definition at line 84 of file qwt_plot.cpp.

References [QwtPlotDict::autoDelete\(\)](#), and [QwtPlotDict::detachItems\(\)](#).

6.49.4 Member Function Documentation

6.49.4.1 void [QwtPlot::setAutoReplot \(bool tf = true\)](#)

Set or reset the autoReplot option.

If the `autoReplot` option is set, the plot will be updated implicitly by manipulating member functions. Since this may be time-consuming, it is recommended to leave this option switched off and call `replot()` explicitly if necessary.

The `autoReplot` option is set to `false` by default, which means that the user has to call `replot()` in order to make changes visible.

Parameters:

tf `true` or `false`. Defaults to `true`.

See also:

[replot\(\)](#)

Definition at line 184 of file `qwt_plot.cpp`.

Referenced by `QwtPlotPrintFilter::apply()`, `QwtPlotCanvas::drawContents()`, `QwtPlotPanner::moveCanvas()`, `replot()`, `QwtPlotZoomer::rescale()`, `QwtPlotMagnifier::rescale()`, and `QwtPlotPrintFilter::reset()`.

6.49.4.2 bool QwtPlot::autoReplot () const**Returns:**

`true` if the `autoReplot` option is set.

Definition at line 190 of file `qwt_plot.cpp`.

Referenced by `QwtPlotPrintFilter::apply()`, `QwtPlotCanvas::drawContents()`, `QwtPlotPanner::moveCanvas()`, `replot()`, `QwtPlotZoomer::rescale()`, `QwtPlotMagnifier::rescale()`, and `QwtPlotPrintFilter::reset()`.

6.49.4.3 void QwtPlot::print (QPaintDevice & *paintDev*, const QwtPlotPrintFilter & *pfilter* = QwtPlotPrintFilter()) const

Print the plot to a `QPaintDevice` (`QPrinter`) This function prints the contents of a `QwtPlot` instance to `QPaintDevice` object. The size is derived from its device metrics.

Parameters:

paintDev device to paint on, often a printer

pfilter print filter

See also:

[QwtPlot::print](#)
[QwtPlotPrintFilter](#)

Definition at line 43 of file `qwt_plot_print.cpp`.

6.49.4.4 void QwtPlot::print (QPainter * *painter*, const QRect & *plotRect*, const QwtPlotPrintFilter & *pfilter* = QwtPlotPrintFilter()) const [virtual]

Paint the plot into a given rectangle. Paint the contents of a `QwtPlot` instance into a given rectangle.

Parameters:

painter Painter
plotRect Bounding rectangle
pfilter Print filter

See also:

[QwtPlotPrintFilter](#)

Definition at line 73 of file qwt_plot_print.cpp.

References [QwtPlotPrintFilter::apply\(\)](#), [axisEnabled\(\)](#), [axisScaleDiv\(\)](#), [axisScaleEngine\(\)](#), [axisWidget\(\)](#), [QwtPlotLayout::canvasMargin\(\)](#), [QwtPlotLayout::canvasRect\(\)](#), [QwtScaleWidget::endBorderDist\(\)](#), [QwtScaleDiv::hBound\(\)](#), [QwtLegend::isEmpty\(\)](#), [QwtScaleDiv::lBound\(\)](#), [legend\(\)](#), [margin\(\)](#), [QwtScaleWidget::margin\(\)](#), [QwtPainter::metricsMap\(\)](#), [QwtPlotPrintFilter::options\(\)](#), [plotLayout\(\)](#), [printCanvas\(\)](#), [printLegend\(\)](#), [printScale\(\)](#), [printTitle\(\)](#), [QwtPlotPrintFilter::reset\(\)](#), [QwtPainter::resetMetricsMap\(\)](#), [QwtPlotLayout::scaleRect\(\)](#), [QwtScaleWidget::setMargin\(\)](#), [QwtPainter::setMetricsMap\(\)](#), [QwtScaleMap::setPaintXInterval\(\)](#), [QwtScaleMap::setScaleInterval\(\)](#), [QwtScaleWidget::startBorderDist\(\)](#), and [titleLabel\(\)](#).

6.49.4.5 [QwtPlotLayout](#) * [QwtPlot::plotLayout](#) ()**Returns:**

the plot's title

Definition at line 228 of file qwt_plot.cpp.

Referenced by [canvasMap\(\)](#), and [print\(\)](#).

6.49.4.6 [const QwtPlotLayout](#) * [QwtPlot::plotLayout](#) () const**Returns:**

the plot's titel label.

Definition at line 234 of file qwt_plot.cpp.

6.49.4.7 [void QwtPlot::setMargin](#) (int *margin*)

Change the margin of the plot. The margin is the space around all components.

Parameters:

margin new margin

See also:

[QwtPlotLayout::setMargin\(\)](#), [margin\(\)](#), [plotLayout\(\)](#)

Definition at line 665 of file qwt_plot.cpp.

References [updateLayout\(\)](#).

6.49.4.8 int QwtPlot::margin () const

Returns:

margin

See also:

[setMargin\(\)](#), [QwtPlotLayout::margin\(\)](#), [plotLayout\(\)](#)

Definition at line 681 of file qwt_plot.cpp.

Referenced by [canvasMap\(\)](#), and [print\(\)](#).

6.49.4.9 void QwtPlot::setTitle (const QString & title)

Change the plot's title

Parameters:

title New title

Definition at line 199 of file qwt_plot.cpp.

References [updateLayout\(\)](#).

Referenced by [applyProperties\(\)](#).

6.49.4.10 void QwtPlot::setTitle (const QwtText & title)

Change the plot's title

Parameters:

title New title

Definition at line 212 of file qwt_plot.cpp.

References [title\(\)](#), and [updateLayout\(\)](#).

6.49.4.11 QwtText QwtPlot::title () const

Returns:

the plot's title

Definition at line 222 of file qwt_plot.cpp.

Referenced by [grabProperties\(\)](#), [QwtPlot\(\)](#), [setAxisTitle\(\)](#), and [setTitle\(\)](#).

6.49.4.12 QwtTextLabel * QwtPlot::titleLabel ()

Returns:

the plot's title label.

Definition at line 240 of file qwt_plot.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#), [print\(\)](#), [printTitle\(\)](#), and [QwtPlotPrintFilter::reset\(\)](#).

6.49.4.13 `const QwtTextLabel * QwtPlot::titleLabel () const`**Returns:**

the plot's titel label.

Definition at line 248 of file qwt_plot.cpp.

6.49.4.14 `QwtPlotCanvas * QwtPlot::canvas ()`**Returns:**

the plot's canvas

Definition at line 275 of file qwt_plot.cpp.

Referenced by `canvasBackground()`, `canvasLineWidth()`, `QwtPlotCurve::draw()`, `QwtPlotLayout::minimumSizeHint()`, `printCanvas()`, `replot()`, `setCanvasBackground()`, and `setCanvasLineWidth()`.

6.49.4.15 `const QwtPlotCanvas * QwtPlot::canvas () const`**Returns:**

the plot's canvas

Definition at line 283 of file qwt_plot.cpp.

6.49.4.16 `void QwtPlot::setCanvasBackground (const QColor & c)`

Change the background of the plotting area.

Sets `c` to `QColorGroup::Background` of all colorgroups of the palette of the canvas. Using `canvas()->setPalette()` is a more powerful way to set these colors.

Parameters:

`c` new background color

Definition at line 694 of file qwt_plot.cpp.

References `canvas()`.

Referenced by `QwtPlotPrintFilter::apply()`, and `QwtPlotPrintFilter::reset()`.

6.49.4.17 `const QColor & QwtPlot::canvasBackground () const`

Nothing else than: `canvas()->palette().color(QPalette::Normal, QColorGroup::Background);`

Returns:

the background color of the plotting area.

Definition at line 716 of file qwt_plot.cpp.

References `canvas()`.

Referenced by `QwtPlotPrintFilter::apply()`.

6.49.4.18 void QwtPlot::setCanvasLineWidth (int *w*)

Change the border width of the plotting area. Nothing else than `canvas()->setLineWidth(w)`, left for compatibility only.

Parameters:

w new border width

Definition at line 733 of file `qwt_plot.cpp`.

References `canvas()`, and `updateLayout()`.

6.49.4.19 int QwtPlot::canvasLineWidth () const

Nothing else than: `canvas()->lineWidth()`, left for compatibility only.

Returns:

the border width of the plotting area

Definition at line 744 of file `qwt_plot.cpp`.

References `canvas()`.

6.49.4.20 QwtScaleMap QwtPlot::canvasMap (int *axisId*) const [virtual]**Parameters:**

axisId Axis

Returns:

Map for the axis on the canvas. With this map pixel coordinates can be translated to plot coordinates and vice versa.

See also:

[QwtScaleMap](#), [transform\(\)](#), [invTransform\(\)](#)

Definition at line 612 of file `qwt_plot.cpp`.

References `axisEnabled()`, `axisScaleDiv()`, `axisScaleEngine()`, `axisWidget()`, `QwtPlotLayout::canvasMargin()`, `QwtScaleWidget::endBorderDist()`, `QwtScaleDiv::hBound()`, `QwtScaleDiv::lBound()`, `margin()`, `plotLayout()`, `QwtScaleMap::setPaintInterval()`, `QwtScaleMap::setScaleInterval()`, `QwtScaleMap::setTransformation()`, and `QwtScaleWidget::startBorderDist()`.

Referenced by `QwtPlotCurve::closestPoint()`, `QwtPlotCurve::draw()`, `drawCanvas()`, `QwtPlotPicker::invTransform()`, `invTransform()`, `QwtPlotPanner::moveCanvas()`, `QwtPlotPicker::transform()`, and `transform()`.

6.49.4.21 double QwtPlot::invTransform (int *axisId*, int *pos*) const

Transform the x or y coordinate of a position in the drawing region into a value.

Parameters:

axisId axis index

pos position

Warning:

The position can be an x or a y coordinate, depending on the specified axis.

Definition at line 349 of file qwt_plot_axis.cpp.

References `axisValid()`, and `canvasMap()`.

6.49.4.22 int QwtPlot::transform (int *axisId*, double *value*) const

Transform a value into a coordinate in the plotting region.

Parameters:

axisId axis index

value value

Returns:

X or y coordinate in the plotting region corresponding to the value.

Definition at line 365 of file qwt_plot_axis.cpp.

References `axisValid()`, and `canvasMap()`.

6.49.4.23 QwtScaleEngine * QwtPlot::axisScaleEngine (int *axisId*)**Returns:**

Scale engine for a specific axis

Definition at line 144 of file qwt_plot_axis.cpp.

References `axisValid()`.

Referenced by `canvasMap()`, and `print()`.

6.49.4.24 const QwtScaleEngine * QwtPlot::axisScaleEngine (int *axisId*) const**Returns:**

Scale engine for a specific axis

Definition at line 153 of file qwt_plot_axis.cpp.

References `axisValid()`.

6.49.4.25 void QwtPlot::setAxisScaleEngine (int *axisId*, QwtScaleEngine * *scaleEngine*)

Change the scale engine for an axis

Parameters:

axisId axis index

scaleEngine Scale engine

See also:

[axisScaleEngine\(\)](#)

Definition at line 128 of file `qwt_plot_axis.cpp`.

References `autoRefresh()`, and `axisValid()`.

6.49.4.26 void QwtPlot::setAxisAutoScale (int *axisId*)

Enable autoscaling for a specified axis.

This member function is used to switch back to autoscaling mode after a fixed scale has been set. Autoscaling is enabled by default.

Parameters:

axisId axis index

See also:

[QwtPlot::setAxisScale\(\)](#), [QwtPlot::setAxisScaleDiv\(\)](#)

Definition at line 396 of file `qwt_plot_axis.cpp`.

References `autoRefresh()`, and `axisValid()`.

6.49.4.27 bool QwtPlot::axisAutoScale (int *axisId*) const

Returns:

`true` if autoscaling is enabled

Parameters:

axisId axis index

Definition at line 164 of file `qwt_plot_axis.cpp`.

References `axisValid()`.

Referenced by `updateAxes()`.

6.49.4.28 void QwtPlot::enableAxis (int *axisId*, bool *tf* = `true`)

Enable or disable a specified axis.

When an axis is disabled, this only means that it is not visible on the screen. Curves, markers and can be attached to disabled axes, and transformation of screen coordinates into values works as normal.

Only `xBottom` and `yLeft` are enabled by default.

Parameters:

axisId axis index

tf `true` (enabled) or `false` (disabled)

Definition at line 332 of file `qwt_plot_axis.cpp`.

References `axisValid()`, and `updateLayout()`.

6.49.4.29 bool QwtPlot::axisEnabled (int *axisId*) const**Returns:**

`true` if a specified axis is enabled

Parameters:

axisId axis index

Definition at line 177 of file `qwt_plot_axis.cpp`.

References `axisValid()`.

Referenced by `canvasMap()`, `QwtPlotLayout::minimumSizeHint()`, `print()`, `printScale()`, `QwtPlotPicker::QwtPlotPicker()`, `sizeHint()`, and `updateLayout()`.

6.49.4.30 void QwtPlot::setAxisFont (int *axisId*, const QFont &*f*)

Change the font of an axis.

Parameters:

axisId axis index

f font

Warning:

This function changes the font of the tick labels, not of the axis title.

Definition at line 381 of file `qwt_plot_axis.cpp`.

References `axisValid()`, and `axisWidget()`.

6.49.4.31 QFont QwtPlot::axisFont (int *axisId*) const**Returns:**

the font of the scale labels for a specified axis

Parameters:

axisId axis index

Definition at line 189 of file `qwt_plot_axis.cpp`.

References `axisValid()`, and `axisWidget()`.

6.49.4.32 void QwtPlot::setAxisScale (int *axisId*, double *min*, double *max*, double *stepSize* = 0)

Disable autoscaling and specify a fixed scale for a selected axis.

Parameters:

axisId axis index

min

max minimum and maximum of the scale

stepSize Major step size. If `step == 0`, the step size is calculated automatically using the max-Major setting.

See also:

[setAxisMaxMajor\(\)](#), [setAxisAutoScale\(\)](#)

Definition at line 414 of file `qwt_plot_axis.cpp`.

References `autoRefresh()`, and `axisValid()`.

Referenced by `QwtPlotPanner::moveCanvas()`, `QwtPlotZoomer::rescale()`, and `QwtPlotMagnifier::rescale()`.

6.49.4.33 void QwtPlot::setAxisScaleDiv (int *axisId*, const [QwtScaleDiv](#) & *scaleDiv*)

Disable autoscaling and specify a fixed scale for a selected axis.

Parameters:

axisId axis index

scaleDiv Scale division

See also:

[setAxisScale\(\)](#), [setAxisAutoScale\(\)](#)

Definition at line 437 of file `qwt_plot_axis.cpp`.

References `autoRefresh()`, and `axisValid()`.

6.49.4.34 void QwtPlot::setAxisScaleDraw (int *axisId*, [QwtScaleDraw](#) * *scaleDraw*)

Set a scale draw.

Parameters:

axisId axis index

scaleDraw object responsible for drawing scales.

By passing `scaleDraw` it is possible to extend [QwtScaleDraw](#) functionality and let it take place in [QwtPlot](#). Please note that `scaleDraw` has to be created with `new` and will be deleted by the corresponding `QwtScale` member (like a child object).

See also:

[QwtScaleDraw](#), [QwtScaleWidget](#)

Warning:

The attributes of `scaleDraw` will be overwritten by those of the previous [QwtScaleDraw](#).

Definition at line 465 of file `qwt_plot_axis.cpp`.

References `autoRefresh()`, `axisValid()`, `axisWidget()`, and `QwtScaleWidget::setScaleDraw()`.

6.49.4.35 double QwtPlot::axisStepSize (int *axisId*) const

Return the step size parameter, that has been set in `setAxisScale`. This doesn't need to be the step size of the current scale.

Parameters:

axisId axis index

Returns:

step size parameter value

See also:

[setAxisScale](#)

Definition at line 300 of file `qwt_plot_axis.cpp`.

References `axisValid()`.

6.49.4.36 const QwtScaleDiv * QwtPlot::axisScaleDiv (int *axisId*) const

Return the scale division of a specified axis.

`axisScaleDiv(axisId)->lBound()`, `axisScaleDiv(axisId)->hBound()` are the current limits of the axis scale.

Parameters:

axisId axis index

Returns:

Scale division

See also:

[QwtScaleDiv](#), [setAxisScaleDiv](#)

Definition at line 235 of file `qwt_plot_axis.cpp`.

References `axisValid()`.

Referenced by `canvasMap()`, `QwtPlotPanner::moveCanvas()`, `print()`, `QwtPlotZoomer::rescale()`, `QwtPlotMagnifier::rescale()`, `QwtPlotPicker::scaleRect()`, `QwtPlotScaleItem::setScaleDivFromAxis()`, `QwtPlotScaleItem::setScaleDraw()`, and `updateAxes()`.

6.49.4.37 QwtScaleDiv * QwtPlot::axisScaleDiv (int *axisId*)

Return the scale division of a specified axis.

`axisScaleDiv(axisId)->lBound()`, `axisScaleDiv(axisId)->hBound()` are the current limits of the axis scale.

Parameters:

axisId axis index

Returns:

Scale division

See also:

[QwtScaleDiv](#), [setAxisScaleDiv](#)

Definition at line 254 of file `qwt_plot_axis.cpp`.

References `axisValid()`.

6.49.4.38 `const QwtScaleDraw * QwtPlot::axisScaleDraw (int axisId) const`

Returns:

the scale draw of a specified axis

Parameters:

axisId axis index

Returns:

specified `scaleDraw` for axis, or `NULL` if axis is invalid.

See also:

[QwtScaleDraw](#)

Definition at line 268 of file `qwt_plot_axis.cpp`.

References `axisValid()`, `axisWidget()`, and `QwtScaleWidget::scaleDraw()`.

6.49.4.39 `QwtScaleDraw * QwtPlot::axisScaleDraw (int axisId)`

Returns:

the scale draw of a specified axis

Parameters:

axisId axis index

Returns:

specified `scaleDraw` for axis, or `NULL` if axis is invalid.

See also:

[QwtScaleDraw](#)

Definition at line 282 of file `qwt_plot_axis.cpp`.

References `axisValid()`, `axisWidget()`, and `QwtScaleWidget::scaleDraw()`.

6.49.4.40 `const QwtScaleWidget * QwtPlot::axisWidget (int axisId) const`

Returns:

specified axis, or `NULL` if `axisId` is invalid.

Parameters:

axisId axis index

Definition at line 100 of file qwt_plot_axis.cpp.

References `axisValid()`.

Referenced by `QwtPlotPrintFilter::apply()`, `axisFont()`, `axisScaleDraw()`, `axisTitle()`, `canvasMap()`, `QwtPlotLayout::minimumSizeHint()`, `print()`, `printScale()`, `QwtPlotPrintFilter::reset()`, `setAxisFont()`, `setAxisLabelRotation()`, `setAxisScaleDraw()`, `setAxisTitle()`, `sizeHint()`, `updateAxes()`, and `updateLayout()`.

6.49.4.41 [QwtScaleWidget](#) * `QwtPlot::axisWidget(int axisId)`**Returns:**

specified axis, or NULL if `axisId` is invalid.

Parameters:

axisId axis index

Definition at line 112 of file qwt_plot_axis.cpp.

References `axisValid()`.

6.49.4.42 `void QwtPlot::setAxisLabelAlignment(int axisId, Qt::Alignment alignment)`

Change the alignment of the tick labels

Parameters:

axisId axis index

alignment Or'd `Qt::AlignmentFlags` <see `qnamespace.h`>

See also:

[QwtScaleDraw::setLabelAlignment\(\)](#)

Definition at line 483 of file qwt_plot_axis.cpp.

6.49.4.43 `void QwtPlot::setAxisLabelRotation(int axisId, double rotation)`

Rotate all tick labels

Parameters:

axisId axis index

rotation Angle in degrees. When changing the label rotation, the label alignment might be adjusted too.

See also:

[QwtScaleDraw::setLabelRotation\(\)](#), [QwtPlot::setAxisLabelAlignment](#)

Definition at line 497 of file qwt_plot_axis.cpp.

References `axisValid()`, `axisWidget()`, and `QwtScaleWidget::setLabelRotation()`.

6.49.4.44 void QwtPlot::setAxisTitle (int *axisId*, const QString & *title*)

Change the title of a specified axis.

Parameters:

axisId axis index

title axis title

Definition at line 561 of file qwt_plot_axis.cpp.

References [axisValid\(\)](#), [axisWidget\(\)](#), and [QwtScaleWidget::setTitle\(\)](#).

6.49.4.45 void QwtPlot::setAxisTitle (int *axisId*, const QwtText & *title*)

Change the title of a specified axis.

Parameters:

axisId axis index

title axis title

Definition at line 572 of file qwt_plot_axis.cpp.

References [axisValid\(\)](#), [axisWidget\(\)](#), [QwtScaleWidget::setTitle\(\)](#), and [title\(\)](#).

6.49.4.46 QwtText QwtPlot::axisTitle (int *axisId*) const**Returns:**

the title of a specified axis

Parameters:

axisId axis index

Definition at line 312 of file qwt_plot_axis.cpp.

References [axisValid\(\)](#), [axisWidget\(\)](#), and [QwtScaleWidget::title\(\)](#).

6.49.4.47 void QwtPlot::setAxisMaxMinor (int *axisId*, int *maxMinor*)

Set the maximum number of minor scale intervals for a specified axis

Parameters:

axisId axis index

maxMinor maximum number of minor steps

See also:

[axisMaxMinor\(\)](#)

Definition at line 510 of file qwt_plot_axis.cpp.

References [autoRefresh\(\)](#), and [axisValid\(\)](#).

6.49.4.48 int QwtPlot::axisMaxMajor (int *axisId*) const**Returns:**

the maximum number of major ticks for a specified axis

Parameters:

axisId axis index sa [setAxisMaxMajor\(\)](#)

Definition at line 203 of file qwt_plot_axis.cpp.

References [axisValid\(\)](#).

6.49.4.49 void QwtPlot::setAxisMaxMajor (int *axisId*, int *maxMajor*)

Set the maximum number of major scale intervals for a specified axis

Parameters:

axisId axis index

maxMajor maximum number of major steps

See also:

[axisMaxMajor\(\)](#)

Definition at line 537 of file qwt_plot_axis.cpp.

References [autoRefresh\(\)](#), and [axisValid\(\)](#).

6.49.4.50 int QwtPlot::axisMaxMinor (int *axisId*) const**Returns:**

the maximum number of minor ticks for a specified axis

Parameters:

axisId axis index sa [setAxisMaxMinor\(\)](#)

Definition at line 216 of file qwt_plot_axis.cpp.

References [axisValid\(\)](#).

6.49.4.51 void QwtPlot::insertLegend (QwtLegend * *legend*, QwtPlot::LegendPosition *pos* = QwtPlot::RightLegend, double *ratio* = -1.0)

Insert a legend.

If the position legend is `QwtPlot::LeftLegend` or `QwtPlot::RightLegend` the legend will be organized in one column from top to down. Otherwise the legend items will be placed in a table with a best fit number of columns from left to right.

If `pos != QwtPlot::ExternalLegend` the plot widget will become parent of the legend. It will be deleted when the plot is deleted, or another legend is set with [insertLegend\(\)](#).

Parameters:

legend Legend

pos The legend's position. For top/left position the number of columns will be limited to 1, otherwise it will be set to unlimited.

ratio Ratio between legend and the bounding rect of title, canvas and axes. The legend will be shrinked if it would need more space than the given ratio. The ratio is limited to]0.0 .. 1.0]. In case of <= 0.0 it will be reset to the default ratio. The default vertical/horizontal ratio is 0.33/0.5.

See also:

[legend\(\)](#), [QwtPlotLayout::legendPosition\(\)](#), [QwtPlotLayout::setLegendPosition\(\)](#)

Definition at line 822 of file qwt_plot.cpp.

References [QwtPlotDict::itemList\(\)](#), [legend\(\)](#), [updateLayout\(\)](#), and [updateTabOrder\(\)](#).

6.49.4.52 [QwtLegend](#) * [QwtPlot::legend](#) ()**Returns:**

the plot's legend

See also:

[insertLegend\(\)](#)

Definition at line 257 of file qwt_plot.cpp.

Referenced by [QwtPlotLayout::activate\(\)](#), [QwtPlotPrintFilter::apply\(\)](#), [insertLegend\(\)](#), [print\(\)](#), [printLegend\(\)](#), and [QwtPlotPrintFilter::reset\(\)](#).

6.49.4.53 [const \[QwtLegend\]\(#\) * \[QwtPlot::legend\]\(#\) \(\) const](#)**Returns:**

the plot's legend

See also:

[insertLegend\(\)](#)

Definition at line 266 of file qwt_plot.cpp.

6.49.4.54 [void \[QwtPlot::polish\]\(#\) \(\)](#) [virtual]

Polish.

Definition at line 289 of file qwt_plot.cpp.

References [replot\(\)](#).

Referenced by [event\(\)](#).

6.49.4.55 `QSize QwtPlot::sizeHint () const` [virtual]

Return sizeHint

See also:

[minimumSizeHint\(\)](#)

Definition at line 303 of file qwt_plot.cpp.

References [axisEnabled\(\)](#), [axisWidget\(\)](#), [minimumSizeHint\(\)](#), [QwtScaleWidget::minimumSizeHint\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), [QwtScaleWidget::scaleDraw\(\)](#), and [QwtScaleDiv::ticks\(\)](#).

6.49.4.56 `QSize QwtPlot::minimumSizeHint () const` [virtual]

Return a minimum size hint.

Definition at line 338 of file qwt_plot.cpp.

Referenced by [sizeHint\(\)](#).

6.49.4.57 `void QwtPlot::updateLayout ()` [virtual]

Adjust plot content to its current size.

See also:

[resizeEvent\(\)](#)

Definition at line 413 of file qwt_plot.cpp.

References [axisEnabled\(\)](#), and [axisWidget\(\)](#).

Referenced by [enableAxis\(\)](#), [event\(\)](#), [insertLegend\(\)](#), [resizeEvent\(\)](#), [setCanvasLineWidth\(\)](#), [setMargin\(\)](#), and [setTitle\(\)](#).

6.49.4.58 `bool QwtPlot::event (QEvent *)` [virtual]

Adds handling of layout requests.

Definition at line 140 of file qwt_plot.cpp.

References [polish\(\)](#), and [updateLayout\(\)](#).

6.49.4.59 `void QwtPlot::legendClicked (QwtPlotItem *plotItem)` [signal]

A signal which is emitted when the user has clicked on a legend item, which is in [QwtLegend::Clickable-Item](#) mode.

Parameters:

plotItem Corresponding plot item of the selected legend item

Note:

clicks are disabled as default

See also:

[QwtLegend::setItemMode](#), [QwtLegend::itemMode](#)

Referenced by [legendItemClicked\(\)](#).

6.49.4.60 void QwtPlot::legendChecked (QwtPlotItem * *plotItem*, bool *on*) [signal]

A signal which is emitted when the user has clicked on a legend item, which is in QwtLegend::Checkable-Item mode

Parameters:

plotItem Corresponding plot item of the selected legend item
on True when the legen item is checked

Note:

clicks are disabled as default

See also:

[QwtLegend::setItemMode](#), [QwtLegend::itemMode](#)

Referenced by [legendItemChecked\(\)](#).

6.49.4.61 void QwtPlot::clear () [virtual, slot]

Remove all curves and markers.

Definition at line 789 of file `qwt_plot.cpp`.

References [QwtPlotDict::detachItems\(\)](#).

6.49.4.62 void QwtPlot::replot () [virtual, slot]

Redraw the plot.

If the `autoReplot` option is not set (which is the default) or if any curves are attached to raw data, the plot has to be refreshed explicitly in order to make changes visible.

See also:

[setAutoReplot\(\)](#)

Warning:

Calls [canvas\(\)->repaint](#), take care of infinite recursions

Definition at line 363 of file `qwt_plot.cpp`.

References [autoReplot\(\)](#), [canvas\(\)](#), [QwtPlotCanvas::invalidatePaintCache\(\)](#), [setAutoReplot\(\)](#), [QwtPlotCanvas::testPaintAttribute\(\)](#), and [updateAxes\(\)](#).

Referenced by [applyProperties\(\)](#), [autoRefresh\(\)](#), [QwtPlotPanner::moveCanvas\(\)](#), [polish\(\)](#), [QwtPlotZoomer::rescale\(\)](#), [QwtPlotMagnifier::rescale\(\)](#), and [QwtPlotZoomer::setZoomBase\(\)](#).

6.49.4.63 void QwtPlot::autoRefresh () [slot]

Replots the plot if [QwtPlot::autoReplot\(\)](#) is `true`.

Definition at line 163 of file `qwt_plot.cpp`.

References [replot\(\)](#).

Referenced by [setAxisAutoScale\(\)](#), [setAxisMaxMajor\(\)](#), [setAxisMaxMinor\(\)](#), [setAxisScale\(\)](#), [setAxisScaleDiv\(\)](#), [setAxisScaleDraw\(\)](#), and [setAxisScaleEngine\(\)](#).

6.49.4.64 void QwtPlot::legendItemClicked () [protected, virtual, slot]

Called internally when the legend has been clicked on. Emits a [legendClicked\(\)](#) signal.

Definition at line 762 of file qwt_plot.cpp.

References [legendClicked\(\)](#).

6.49.4.65 void QwtPlot::legendItemChecked (bool on) [protected, virtual, slot]

Called internally when the legend has been checked Emits a [legendClicked\(\)](#) signal.

Definition at line 777 of file qwt_plot.cpp.

References [legendChecked\(\)](#).

6.49.4.66 bool QwtPlot::axisValid (int axisId) [static, protected]**Returns:**

true if the specified axis exists, otherwise false

Parameters:

axisId axis index

Definition at line 753 of file qwt_plot.cpp.

Referenced by [axisAutoScale\(\)](#), [axisEnabled\(\)](#), [axisFont\(\)](#), [axisMaxMajor\(\)](#), [axisMaxMinor\(\)](#), [axisScaleDiv\(\)](#), [axisScaleDraw\(\)](#), [axisScaleEngine\(\)](#), [axisStepSize\(\)](#), [axisTitle\(\)](#), [axisWidget\(\)](#), [enableAxis\(\)](#), [invertTransform\(\)](#), [setAxisAutoScale\(\)](#), [setAxisFont\(\)](#), [setAxisLabelRotation\(\)](#), [setAxisMaxMajor\(\)](#), [setAxisMaxMinor\(\)](#), [setAxisScale\(\)](#), [setAxisScaleDiv\(\)](#), [setAxisScaleDraw\(\)](#), [setAxisScaleEngine\(\)](#), [setAxisTitle\(\)](#), and [transform\(\)](#).

6.49.4.67 void QwtPlot::drawCanvas (QPainter *painter) [protected, virtual]

Redraw the canvas.

Parameters:

painter Painter used for drawing

Warning:

[drawCanvas](#) calls [drawItems](#) what is also used for printing. Applications that like to add individual plot items better overload [drawItems\(\)](#)

See also:

[drawItems\(\)](#)

Definition at line 554 of file qwt_plot.cpp.

References [canvasMap\(\)](#), and [drawItems\(\)](#).

6.49.4.68 void QwtPlot::drawItems (QPainter *painter, const QRect &rect, const QwtScaleMap map[axisCnt], const QwtPlotPrintFilter &filter) const [protected, virtual]

Redraw the canvas items.

Parameters:

- painter* Painter used for drawing
- rect* Bounding rectangle where to paint
- map* QwtPlot::axisCnt maps, mapping between plot and paint device coordinates
- pfilter* Plot print filter

Definition at line 572 of file qwt_plot.cpp.

References QwtPlotDict::itemList(), and QwtPlotPrintFilter::options().

Referenced by drawCanvas(), and printCanvas().

6.49.4.69 void QwtPlot::updateTabOrder () [protected, virtual]

Update the focus tab order

The order is changed so that the canvas will be in front of the first legend item, or behind the last legend item - depending on the position of the legend.

Definition at line 477 of file qwt_plot.cpp.

Referenced by insertLegend().

6.49.4.70 void QwtPlot::updateAxes () [protected]

Rebuild the scales.

Definition at line 579 of file qwt_plot_axis.cpp.

References axisAutoScale(), axisScaleDiv(), axisWidget(), QwtScaleWidget::getBorderDistHint(), QwtDoubleInterval::isValid(), QwtPlotDict::itemList(), QwtDoubleInterval::maxValue(), QwtDoubleInterval::minValue(), QwtScaleWidget::setBorderDist(), and QwtScaleWidget::setScaleDiv().

Referenced by replot().

6.49.4.71 void QwtPlot::resizeEvent (QResizeEvent * e) [protected, virtual]

Resize and update internal layout.

Definition at line 347 of file qwt_plot.cpp.

References updateLayout().

6.49.4.72 void QwtPlot::printLegendItem (QPainter * painter, const QWidget * w, const QRect & rect) const [protected, virtual]

Print the legend item into a given rectangle.

Parameters:

- painter* Painter
- w* Widget representing a legend item
- rect* Bounding rectangle

Definition at line 337 of file qwt_plot_print.cpp.

Referenced by printLegend().

6.49.4.73 void QwtPlot::printTitle (QPainter * *painter*, const QRect & *rect*) const [protected, virtual]

Print the title into a given rectangle.

Parameters:

painter Painter

rect Bounding rectangle

Definition at line 258 of file qwt_plot_print.cpp.

References QwtText::draw(), QwtTextLabel::text(), and titleLabel().

Referenced by print().

6.49.4.74 void QwtPlot::printScale (QPainter * *painter*, int *axisId*, int *startDist*, int *endDist*, int *baseDist*, const QRect & *rect*) const [protected, virtual]

Paint a scale into a given rectangle. Paint the scale into a given rectangle.

Parameters:

painter Painter

axisId Axis

startDist Start border distance

endDist End border distance

baseDist Base distance

rect Bounding rectangle

Definition at line 361 of file qwt_plot_print.cpp.

References axisEnabled(), axisWidget(), QwtScaleWidget::colorBarRect(), QwtScaleWidget::colorBarWidth(), QwtAbstractScaleDraw::draw(), QwtScaleWidget::drawColorBar(), QwtScaleWidget::drawTitle(), QwtScaleWidget::isColorBarEnabled(), QwtMetricsMap::layoutToScreen(), QwtScaleDraw::length(), QwtPainter::metricsMap(), QwtScaleDraw::move(), QwtScaleDraw::orientation(), QwtScaleWidget::penWidth(), QwtScaleDraw::pos(), QwtScaleWidget::scaleDraw(), QwtScaleDraw::setLength(), and QwtScaleWidget::spacing().

Referenced by print().

6.49.4.75 void QwtPlot::printCanvas (QPainter * *painter*, const QRect & *boundingRect*, const QRect & *canvasRect*, const QwtScaleMap *map*[axisCnt], const QwtPlotPrintFilter & *pfilter*) const [protected, virtual]

Print the canvas into a given rectangle.

Parameters:

painter Painter

map Maps mapping between plot and paint device coordinates

boundingRect Bounding rectangle

canvasRect Canvas rectangle

pfilter Print filter

See also:

[QwtPlotPrintFilter](#)

Definition at line 469 of file qwt_plot_print.cpp.

References [canvas\(\)](#), [drawItems\(\)](#), [QwtPainter::drawRect\(\)](#), [QwtPainter::fillRect\(\)](#), [QwtPlotPrintFilter::options\(\)](#), and [QwtPainter::setClipRect\(\)](#).

Referenced by [print\(\)](#).

6.49.4.76 void QwtPlot::printLegend (QPainter * *painter*, const QRect & *rect*) const
[protected, virtual]

Print the legend into a given rectangle.

Parameters:

painter Painter

rect Bounding rectangle

Definition at line 282 of file qwt_plot_print.cpp.

References [QwtDynGridLayout::columnsForWidth\(\)](#), [QwtLegend::contentsWidget\(\)](#), [QwtDynGridLayout::count\(\)](#), [QwtDynGridLayout::itemAt\(\)](#), [QwtDynGridLayout::layoutItems\(\)](#), [legend\(\)](#), [printLegendItem\(\)](#), and [QwtPainter::setClipRect\(\)](#).

Referenced by [print\(\)](#).

6.50 QwtPlotCanvas Class Reference

6.50.1 Detailed Description

Canvas of a [QwtPlot](#).

See also:

[QwtPlot](#)

Definition at line 26 of file qwt_plot_canvas.h.

Public Types

- enum [PaintAttribute](#) {
 PaintCached = 1,
 PaintPacked = 2,
 PaintFiltered = 1,
 ClipPolygons = 2,
 PaintUsingTextFont = 1,
 PaintUsingTextColor = 2,
 PaintBackground = 4 }

- enum [FocusIndicator](#) {
NoFocusIndicator,
CanvasFocusIndicator,
ItemFocusIndicator }

Public Member Functions

- [QwtPlotCanvas](#) ([QwtPlot](#) *)
- virtual [~QwtPlotCanvas](#) ()
- [QwtPlot](#) * [plot](#) ()
- const [QwtPlot](#) * [plot](#) () const
- void [setFocusIndicator](#) ([FocusIndicator](#))
- [FocusIndicator](#) [focusIndicator](#) () const
- void [setPaintAttribute](#) ([PaintAttribute](#), bool on=true)
- bool [testPaintAttribute](#) ([PaintAttribute](#)) const
- QPixmap * [paintCache](#) ()
- const QPixmap * [paintCache](#) () const
- void [invalidatePaintCache](#) ()

Protected Member Functions

- virtual void [hideEvent](#) (QHideEvent *)
- virtual void [paintEvent](#) (QPaintEvent *)
- virtual void [drawContents](#) (QPainter *)
- virtual void [drawFocusIndicator](#) (QPainter *)
- void [drawCanvas](#) (QPainter *painter=NULL)

6.50.2 Member Enumeration Documentation

6.50.2.1 enum [QwtPlotCanvas::PaintAttribute](#)

Paint attributes.

- **PaintCached**
Paint double buffered and reuse the content of the pixmap buffer for some spontaneous repaints that happen when a plot gets unhidden, deiconified or changes the focus. Disabling the cache will improve the performance for incremental paints (using [QwtPlotCurve::draw](#)).
- **PaintPacked**
Suppress system background repaints and paint it together with the canvas contents. Painting packed might avoid flickering for expensive repaints, when there is a notable gap between painting the background and the plot contents.

The default setting enables PaintCached and PaintPacked

See also:

[setPaintAttribute\(\)](#), [testPaintAttribute\(\)](#), [paintCache\(\)](#)

Definition at line 53 of file `qwt_plot_canvas.h`.

6.50.2.2 enum `QwtPlotCanvas::FocusIndicator`

Focus indicator.

- `NoFocusIndicator`
Don't paint a focus indicator
- `CanvasFocusIndicator`
The focus is related to the complete canvas. Paint the focus indicator using `paintFocus()`
- `ItemFocusIndicator`
The focus is related to an item (curve, point, ...) on the canvas. It is up to the application to display a focus indication using f.e. highlighting.

See also:

`setFocusIndicator()`, `focusIndicator()`, `paintFocus()`

Definition at line 77 of file `qwt_plot_canvas.h`.

6.50.3 Constructor & Destructor Documentation

6.50.3.1 `QwtPlotCanvas::QwtPlotCanvas (QwtPlot *)` `[explicit]`

Sets a cross cursor, enables `QwtPlotCanvas::PaintCached`.

Definition at line 50 of file `qwt_plot_canvas.cpp`.

References `setPaintAttribute()`.

6.50.3.2 `QwtPlotCanvas::~~QwtPlotCanvas ()` `[virtual]`

Destructor.

Definition at line 75 of file `qwt_plot_canvas.cpp`.

6.50.4 Member Function Documentation

6.50.4.1 `QwtPlot * QwtPlotCanvas::plot ()`

Return parent plot widget.

Definition at line 81 of file `qwt_plot_canvas.cpp`.

Referenced by `drawContents()`.

6.50.4.2 `const QwtPlot * QwtPlotCanvas::plot () const`

Return parent plot widget.

Definition at line 91 of file `qwt_plot_canvas.cpp`.

6.50.4.3 void QwtPlotCanvas::setFocusIndicator ([FocusIndicator](#) *focusIndicator*)

Set the focus indicator

See also:

[FocusIndicator](#), [focusIndicator](#)

Definition at line 196 of file qwt_plot_canvas.cpp.

6.50.4.4 [QwtPlotCanvas::FocusIndicator](#) QwtPlotCanvas::focusIndicator () const

Returns:

Focus indicator

See also:

[FocusIndicator](#), [setFocusIndicator](#)

Definition at line 206 of file qwt_plot_canvas.cpp.

Referenced by [drawContents\(\)](#).

6.50.4.5 void QwtPlotCanvas::setPaintAttribute ([PaintAttribute](#) *attribute*, bool *on* = true)

Changing the paint attributes.

Parameters:

attribute Paint attribute

on On/Off

The default setting enables PaintCached and PaintPacked

See also:

[testPaintAttribute\(\)](#), [drawCanvas\(\)](#), [drawContents\(\)](#), [paintCache\(\)](#)

Definition at line 110 of file qwt_plot_canvas.cpp.

Referenced by [QwtPlotCanvas\(\)](#).

6.50.4.6 bool QwtPlotCanvas::testPaintAttribute ([PaintAttribute](#) *attribute*) const

Test wether a paint attribute is enabled

Parameters:

attribute Paint attribute

Returns:

true if the attribute is enabled

See also:

[setPaintAttribute\(\)](#)

Definition at line 167 of file qwt_plot_canvas.cpp.

Referenced by [QwtPlotCurve::draw\(\)](#), and [QwtPlot::replot\(\)](#).

6.50.4.7 QPixmap * QwtPlotCanvas::paintCache ()

Return the paint cache, might be null.

Definition at line 173 of file qwt_plot_canvas.cpp.

Referenced by QwtPlotCurve::draw().

6.50.4.8 const QPixmap * QwtPlotCanvas::paintCache () const

Return the paint cache, might be null.

Definition at line 179 of file qwt_plot_canvas.cpp.

6.50.4.9 void QwtPlotCanvas::invalidatePaintCache ()

Invalidate the internal paint cache.

Definition at line 185 of file qwt_plot_canvas.cpp.

Referenced by QwtPlot::replot().

6.50.4.10 void QwtPlotCanvas::paintEvent (QPaintEvent *) [protected, virtual]

Paint event.

Definition at line 225 of file qwt_plot_canvas.cpp.

References drawContents().

6.50.4.11 void QwtPlotCanvas::drawContents (QPainter *) [protected, virtual]

Redraw the canvas, and focus rect.

Definition at line 250 of file qwt_plot_canvas.cpp.

References QwtPlot::autoReplot(), drawCanvas(), drawFocusIndicator(), focusIndicator(), plot(), and QwtPlot::setAutoReplot().

Referenced by paintEvent().

6.50.4.12 void QwtPlotCanvas::drawFocusIndicator (QPainter *) [protected, virtual]

Draw the focus indication.

Definition at line 350 of file qwt_plot_canvas.cpp.

References QwtPainter::drawFocusRect().

Referenced by drawContents().

6.50.4.13 void QwtPlotCanvas::drawCanvas (QPainter * *painter* = NULL) [protected]

Draw the the canvas

Paints all plot items to the contentsRect(), using [QwtPlot::drawCanvas](#) and updates the paint cache.

See also:

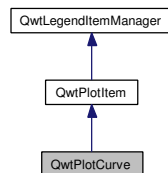
[QwtPlot::drawCanvas](#), [setPaintAttributes\(\)](#), [testPaintAttributes\(\)](#)

Definition at line 281 of file qwt_plot_canvas.cpp.

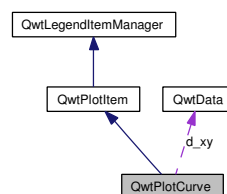
Referenced by drawContents().

6.51 QwtPlotCurve Class Reference

Inheritance diagram for QwtPlotCurve:



Collaboration diagram for QwtPlotCurve:



6.51.1 Detailed Description

A class which draws curves.

This class can be used to display data as a curve in the x-y plane. It supports different display styles, spline interpolation and symbols.

Usage

A. Assign curve properties When a curve is created, it is configured to draw black solid lines with `QwtPlotCurve::Lines` and no symbols. You can change this by calling `setPen()`, `setStyle()` and `setSymbol()`.

B. Assign or change data. Data can be set in two ways:

- `setData()` is overloaded to initialize the x and y data by copying from different data structures with different kind of copy semantics.
- `setRawData()` only stores the pointers and size information and is provided for backwards compatibility. This function is less safe (you must not delete the data while they are attached), but has been more efficient, and has been more convenient for dynamically changing data. Use of `setData()` in combination with a problem-specific subclass of `QwtData` is always preferable.

C. Draw `draw()` maps the data into pixel coordinates and paints them.

Example:

see examples/curvdemo

See also:

[QwtData](#), [QwtSymbol](#), [QwtScaleMap](#)

Definition at line 56 of file qwt_plot_curve.h.

Public Types

- enum [CurveType](#) {
 Yfx,
 Xfy }
- enum [CurveStyle](#) {
 NoCurve,
 Lines,
 Sticks,
 Steps,
 Dots,
 UserCurve = 100 }
- enum [CurveAttribute](#) {
 Inverted = 1,
 Fitted = 2 }
- enum [PaintAttribute](#) {
 PaintCached = 1,
 PaintPacked = 2,
 PaintFiltered = 1,
 ClipPolygons = 2,
 PaintUsingTextFont = 1,
 PaintUsingTextColor = 2,
 PaintBackground = 4 }

Public Member Functions

- [QwtPlotCurve](#) ()
- [QwtPlotCurve](#) (const [QwtText](#) &title)
- [QwtPlotCurve](#) (const QString &title)
- virtual [~QwtPlotCurve](#) ()
- virtual int [rtti](#) () const
- void [setCurveType](#) ([CurveType](#))
- [CurveType](#) [curveType](#) () const
- void [setPaintAttribute](#) ([PaintAttribute](#), bool on=true)
- bool [testPaintAttribute](#) ([PaintAttribute](#)) const
- void [setRawData](#) (const double *x, const double *y, int size)
- void [setData](#) (const double *xData, const double *yData, int size)
- void [setData](#) (const QwtArray< double > &xData, const QwtArray< double > &yData)
- void [setData](#) (const QPolygonF &data)
- void [setData](#) (const [QwtData](#) &data)

- `int closestPoint (const QPoint &pos, double *dist=NULL) const`
- `QwtData & data ()`
- `const QwtData & data () const`
- `int dataSize () const`
- `double x (int i) const`
- `double y (int i) const`
- `virtual QwtDoubleRect boundingRect () const`
- `double minXValue () const`
- `double maxXValue () const`
- `double minYValue () const`
- `double maxYValue () const`
- `void setCurveAttribute (CurveAttribute, bool on=true)`
- `bool testCurveAttribute (CurveAttribute) const`
- `void setPen (const QPen &)`
- `const QPen & pen () const`
- `void setBrush (const QBrush &)`
- `const QBrush & brush () const`
- `void setBaseline (double ref)`
- `double baseline () const`
- `void setStyle (CurveStyle style)`
- `CurveStyle style () const`
- `void setSymbol (const QwtSymbol &s)`
- `const QwtSymbol & symbol () const`
- `void setCurveFitter (QwtCurveFitter *)`
- `QwtCurveFitter * curveFitter () const`
- `virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRect &) const`
- `virtual void draw (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void draw (int from, int to) const`
- `virtual void updateLegend (QwtLegend *) const`

Protected Member Functions

- `void init ()`
- `virtual void drawCurve (QPainter *p, int style, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `virtual void drawSymbols (QPainter *p, const QwtSymbol &, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void drawLines (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void drawSticks (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void drawDots (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void drawSteps (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const`
- `void fillCurve (QPainter *, const QwtScaleMap &, const QwtScaleMap &, QwtPolygon &) const`
- `void closePolyline (const QwtScaleMap &, const QwtScaleMap &, QwtPolygon &) const`

6.51.2 Member Enumeration Documentation

6.51.2.1 enum [QwtPlotCurve::CurveStyle](#)

Curve styles.

See also:

[setStyle](#)

Definition at line 69 of file qwt_plot_curve.h.

6.51.2.2 enum [QwtPlotCurve::CurveAttribute](#)

Curve attributes.

See also:

[setCurveAttribute](#), [testCurveAttribute](#)

Definition at line 85 of file qwt_plot_curve.h.

6.51.2.3 enum [QwtPlotCurve::PaintAttribute](#)

Paint attributes

See also:

[setPaintAttribute](#), [testPaintAttribute](#)

Definition at line 95 of file qwt_plot_curve.h.

6.51.3 Constructor & Destructor Documentation

6.51.3.1 [QwtPlotCurve::QwtPlotCurve \(\)](#) [explicit]

Constructor.

Definition at line 257 of file qwt_plot_curve.cpp.

References [init\(\)](#).

6.51.3.2 [QwtPlotCurve::QwtPlotCurve \(const \[QwtText\]\(#\) & title\)](#) [explicit]

Constructor

Parameters:

title title of the curve

Definition at line 267 of file qwt_plot_curve.cpp.

References [init\(\)](#).

6.51.3.3 [QwtPlotCurve::QwtPlotCurve \(const QString & title\)](#) [explicit]

Constructor

Parameters:

title title of the curve

Definition at line 277 of file qwt_plot_curve.cpp.

References `init()`.

6.51.3.4 QwtPlotCurve::~QwtPlotCurve () [virtual]

Destructor.

Definition at line 284 of file qwt_plot_curve.cpp.

6.51.4 Member Function Documentation**6.51.4.1 int QwtPlotCurve::rtti () const [virtual]****Returns:**

QwtPlotItem::Rtti_PlotCurve

Reimplemented from [QwtPlotItem](#).

Definition at line 305 of file qwt_plot_curve.cpp.

6.51.4.2 void QwtPlotCurve::setCurveType (CurveType curveType)

Assign the curve type

QwtPlotCurve::Yfx Draws y as a function of x (the default). The baseline is interpreted as a horizontal line with y = [baseline\(\)](#).

QwtPlotCurve::Xfy Draws x as a function of y. The baseline is interpreted as a vertical line with x = [baseline\(\)](#).

The baseline is used for aligning the sticks, or filling the curve with a brush.

See also:

[curveType\(\)](#)

Definition at line 1130 of file qwt_plot_curve.cpp.

References `QwtPlotItem::itemChanged()`.

6.51.4.3 QwtPlotCurve::CurveType QwtPlotCurve::curveType () const

Return the curve type

See also:

[setCurveType\(\)](#)

Definition at line 1143 of file qwt_plot_curve.cpp.

6.51.4.4 void QwtPlotCurve::setPaintAttribute (PaintAttribute attribute, bool on = true)

Specify an attribute how to draw the curve.

The attributes can be used to modify the drawing algorithm.

The following attributes are defined:

PaintFiltered Tries to reduce the data that has to be painted, by sorting out duplicates, or paintings outside the visible area. Might have a notable impact on curves with many close points. Only a couple of very basic filtering algos are implemented.

ClipPolygons Clip polygons before painting them. In situations, where points are far outside the visible area this might be a great improvement for the painting performance (especially on Windows).

The default is, that no paint attributes are enabled.

Parameters:

attribute Paint attribute

on On/Off /sa [testPaintAttribute\(\)](#)

Definition at line 333 of file `qwt_plot_curve.cpp`.

6.51.4.5 bool QwtPlotCurve::testPaintAttribute (PaintAttribute attribute) const

Return the current paint attributes.

See also:

[setPaintAttribute](#)

Definition at line 345 of file `qwt_plot_curve.cpp`.

6.51.4.6 void QwtPlotCurve::setRawData (const double * xData, const double * yData, int size)

Initialize the data by pointing to memory blocks which are not managed by [QwtPlotCurve](#).

`setRawData` is provided for efficiency. It is important to keep the pointers during the lifetime of the underlying [QwtCPointerData](#) class.

Parameters:

xData pointer to x data

yData pointer to y data

size size of x and y

See also:

[QwtCPointerData::setData](#).

Definition at line 547 of file `qwt_plot_curve.cpp`.

References [QwtPlotItem::itemChanged\(\)](#).

6.51.4.7 void QwtPlotCurve::setData (const double * *xData*, const double * *yData*, int *size*)

Set data by copying x- and y-values from specified memory blocks. Contrary to setCurveRawData(), this function makes a 'deep copy' of the data.

Parameters:

xData pointer to x values
yData pointer to y values
size size of xData and yData

See also:

[QwtCPointerData](#)

Definition at line 481 of file qwt_plot_curve.cpp.

References QwtPlotItem::itemChanged().

6.51.4.8 void QwtPlotCurve::setData (const QwtArray< double > & *xData*, const QwtArray< double > & *yData*)

Initialize data with x- and y-arrays (explicitly shared).

Parameters:

xData x data
yData y data

See also:

[QwtArrayData](#)

Definition at line 496 of file qwt_plot_curve.cpp.

References QwtPlotItem::itemChanged().

6.51.4.9 void QwtPlotCurve::setData (const QPolygonF & *data*)

Initialize data with an array of points (explicitly shared).

Parameters:

data Data

See also:

[QwtPolygonFData](#)

Definition at line 513 of file qwt_plot_curve.cpp.

6.51.4.10 void QwtPlotCurve::setData (const [QwtData](#) & *data*)

Initialize data with a pointer to [QwtData](#).

Parameters:

data Data

See also:

[QwtData::copy\(\)](#)

Definition at line 527 of file qwt_plot_curve.cpp.

References [QwtData::copy\(\)](#), [data\(\)](#), and [QwtPlotItem::itemChanged\(\)](#).

6.51.4.11 [QwtData](#) & [QwtPlotCurve::data](#) () [\[inline\]](#)**Returns:**

the the curve data

Definition at line 217 of file qwt_plot_curve.h.

Referenced by [setData\(\)](#).

6.51.4.12 [const QwtData](#) & [QwtPlotCurve::data](#) () [const](#) [\[inline\]](#)**Returns:**

the the curve data

Definition at line 223 of file qwt_plot_curve.h.

6.51.4.13 [int](#) [QwtPlotCurve::dataSize](#) () [const](#)

Return the size of the data arrays

See also:

[setData\(\)](#)

Definition at line 1332 of file qwt_plot_curve.cpp.

References [QwtData::size\(\)](#).

Referenced by [closestPoint\(\)](#), [draw\(\)](#), and [drawCurve\(\)](#).

6.51.4.14 [double](#) [QwtPlotCurve::x](#) ([int i](#)) [const](#) [\[inline\]](#)**Parameters:**

i index

Returns:

x-value at position i

Definition at line 232 of file qwt_plot_curve.h.

References [QwtData::x\(\)](#).

Referenced by [closestPoint\(\)](#), [drawDots\(\)](#), [drawLines\(\)](#), [drawSteps\(\)](#), [drawSticks\(\)](#), and [drawSymbols\(\)](#).

6.51.4.15 `double QwtPlotCurve::y (int i) const` `[inline]`**Parameters:**

i index

Returns:

y-value at position *i*

Definition at line 241 of file `qwt_plot_curve.h`.

References `QwtData::y()`.

Referenced by `closestPoint()`, `drawDots()`, `drawLines()`, `drawSteps()`, `drawSticks()`, and `drawSymbols()`.

6.51.4.16 `QwtDoubleRect QwtPlotCurve::boundingRect () const` `[virtual]`

Returns the bounding rectangle of the curve data. If there is no bounding rect, like for empty data the rectangle is invalid.

See also:

[QwtData::boundingRect\(\)](#), [QwtDoubleRect::isValid\(\)](#)

Reimplemented from [QwtPlotItem](#).

Definition at line 560 of file `qwt_plot_curve.cpp`.

References `QwtData::boundingRect()`.

6.51.4.17 `double QwtPlotCurve::minXValue () const` `[inline]`

[boundingRect\(\).left\(\)](#)

Definition at line 137 of file `qwt_plot_curve.h`.

References `QwtPlotItem::boundingRect()`.

6.51.4.18 `double QwtPlotCurve::maxXValue () const` `[inline]`

[boundingRect\(\).right\(\)](#)

Definition at line 139 of file `qwt_plot_curve.h`.

References `QwtPlotItem::boundingRect()`.

6.51.4.19 `double QwtPlotCurve::minYValue () const` `[inline]`

[boundingRect\(\).top\(\)](#)

Definition at line 141 of file `qwt_plot_curve.h`.

References `QwtPlotItem::boundingRect()`.

6.51.4.20 `double QwtPlotCurve::maxYValue () const` `[inline]`

[boundingRect\(\).bottom\(\)](#)

Definition at line 143 of file `qwt_plot_curve.h`.

References `QwtPlotItem::boundingRect()`.

6.51.4.21 void QwtPlotCurve::setCurveAttribute (CurveAttribute attribute, bool on = true)

Specify an attribute for drawing the curve.

The attributes can be used to modify the drawing style. The following attributes are defined:

Fitted For Lines only. A [QwtCurveFitter](#) tries to interpolate/smooth the curve, before it is painted. Note that curve fitting requires temporary memory for calculating coefficients and additional points. If painting in Fitted mode is slow it might be better to fit the points, before they are passed to [QwtPlotCurve](#).

Inverted For Steps only. Draws a step function from the right to the left.

Parameters:

attribute Curve attribute

on On/Off

/sa [testCurveAttribute\(\)](#), [setCurveFitter\(\)](#)

Definition at line 1092 of file qwt_plot_curve.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.51.4.22 bool QwtPlotCurve::testCurveAttribute (CurveAttribute attribute) const**Returns:**

true, if attribute is enabled

See also:

[setCurveAttribute\(\)](#)

Definition at line 1109 of file qwt_plot_curve.cpp.

Referenced by [drawCurve\(\)](#).

6.51.4.23 void QwtPlotCurve::setPen (const QPen & pen)

Assign a pen.

Parameters:

pen New pen

See also:

[pen\(\)](#), [brush\(\)](#)

Definition at line 421 of file qwt_plot_curve.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

Referenced by [QwtPlotPrintFilter::apply\(\)](#), and [QwtPlotPrintFilter::reset\(\)](#).

6.51.4.24 const QPen & QwtPlotCurve::pen () const

Return the pen used to draw the lines.

See also:

[setPen\(\)](#), [brush\(\)](#)

Definition at line 434 of file `qwt_plot_curve.cpp`.

Referenced by `QwtPlotPrintFilter::apply()`, `draw()`, `drawSymbols()`, `QwtPlotPrintFilter::reset()`, and `updateLegend()`.

6.51.4.25 void QwtPlotCurve::setBrush (const QBrush & brush)

Assign a brush. In case of `brush.style() != QBrush::NoBrush` and [style\(\)](#) `!= QwtPlotCurve::Sticks` the area between the curve and the baseline will be filled. In case `!brush.color().isValid()` the area will be filled by `pen.color()`. The fill algorithm simply connects the first and the last curve point to the baseline. So the curve data has to be sorted (ascending or descending).

Parameters:

brush New brush

See also:

[brush\(\)](#), [setBaseline\(\)](#), [baseline\(\)](#)

Definition at line 451 of file `qwt_plot_curve.cpp`.

References `QwtPlotItem::itemChanged()`.

6.51.4.26 const QBrush & QwtPlotCurve::brush () const

Return the brush used to fill the area between lines and the baseline.

See also:

[setBrush\(\)](#), [setBaseline\(\)](#), [baseline\(\)](#)

Definition at line 464 of file `qwt_plot_curve.cpp`.

6.51.4.27 void QwtPlotCurve::setBaseline (double reference)

Set the value of the baseline.

The baseline is needed for filling the curve with a brush or the Sticks drawing style. The default value is 0.0. The interpretation of the baseline depends on the CurveType. With `QwtPlotCurve::Yfx`, the baseline is interpreted as a horizontal line at `y = baseline\(\)`, with `QwtPlotCurve::Yfy`, it is interpreted as a vertical line at `x = baseline\(\)`.

Parameters:

reference baseline

See also:

[baseline\(\)](#), [setBrush\(\)](#), [setStyle\(\)](#), [setCurveType\(\)](#)

Definition at line 1310 of file qwt_plot_curve.cpp.

References `QwtPlotItem::itemChanged()`.

6.51.4.28 `double QwtPlotCurve::baseline () const`

Return the value of the baseline

See also:

[setBaseline](#)

Definition at line 1323 of file qwt_plot_curve.cpp.

6.51.4.29 `void QwtPlotCurve::setStyle (CurveStyle style)`

Set the curve's drawing style.

Valid styles are:

NoCurve Don't draw a curve. Note: This doesn't affect the symbol.

Lines Connect the points with straight lines. The lines might be interpolated depending on the 'Fitted' option. Curve fitting can be configured using `setCurveFitter`.

Sticks Draw vertical sticks from a baseline which is defined by [setBaseline\(\)](#).

Steps Connect the points with a step function. The step function is drawn from the left to the right or vice versa, depending on the 'Inverted' option.

Dots Draw dots at the locations of the data points. Note: This is different from a dotted line (see [setPen\(\)](#)).

UserCurve ... Styles \geq UserCurve are reserved for derived classes of `QwtPlotCurve` that overload [drawCurve\(\)](#) with additional application specific curve types.

See also:

[style\(\)](#)

Definition at line 377 of file qwt_plot_curve.cpp.

References `QwtPlotItem::itemChanged()`.

6.51.4.30 `QwtPlotCurve::CurveStyle QwtPlotCurve::style () const`

Return the current style.

See also:

[setStyle\(\)](#)

Definition at line 390 of file qwt_plot_curve.cpp.

Referenced by `updateLegend()`.

6.51.4.31 void QwtPlotCurve::setSymbol (const [QwtSymbol](#) & *symbol*)

Assign a symbol.

Parameters:

symbol Symbol

See also:

[symbol\(\)](#)

Definition at line 400 of file qwt_plot_curve.cpp.

References [QwtSymbol::clone\(\)](#), [QwtPlotItem::itemChanged\(\)](#), and [symbol\(\)](#).

Referenced by [QwtPlotPrintFilter::apply\(\)](#), and [QwtPlotPrintFilter::reset\(\)](#).

6.51.4.32 const [QwtSymbol](#) & QwtPlotCurve::symbol () const

Return the current symbol.

See also:

[setSymbol\(\)](#)

Definition at line 411 of file qwt_plot_curve.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#), [drawSymbols\(\)](#), [QwtPlotPrintFilter::reset\(\)](#), [setSymbol\(\)](#), and [updateLegend\(\)](#).

6.51.4.33 void QwtPlotCurve::draw (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, const QRect &) const [virtual]

Draw the complete curve.

Parameters:

painter Painter

xMap Maps x-values into pixel coordinates.

yMap Maps y-values into pixel coordinates.

See also:

[drawCurve\(\)](#), [drawSymbols\(\)](#)

Implements [QwtPlotItem](#).

Definition at line 577 of file qwt_plot_curve.cpp.

Referenced by [draw\(\)](#).

6.51.4.34 void QwtPlotCurve::draw (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, int *from*, int *to*) const [virtual]

Draw an interval of the curve.

Parameters:*painter* Painter*xMap* maps x-values into pixel coordinates.*yMap* maps y-values into pixel coordinates.*from* index of the first point to be painted*to* index of the last point to be painted. If *to* < 0 the curve will be painted to its last point.**See also:**[drawCurve\(\)](#), [drawSymbols\(\)](#),

Definition at line 687 of file qwt_plot_curve.cpp.

References [dataSize\(\)](#), [drawCurve\(\)](#), [drawSymbols\(\)](#), [QwtPainter::metricsMap\(\)](#), and [pen\(\)](#).**6.51.4.35 void QwtPlotCurve::draw (int *from*, int *to*) const**

Draw a set of points of a curve.

When observing an measurement while it is running, new points have to be added to an existing curve. [drawCurve](#) can be used to display them avoiding a complete redraw of the canvas.Setting [plot\(\)](#)->canvas()->setAttribute(Qt::WA_PaintOutsidePaintEvent, true); will result in faster painting, if the paint engine of the canvas widget supports this feature.**Parameters:***from* Index of the first point to be painted*to* Index of the last point to be painted. If *to* < 0 the curve will be painted to its last point.**See also:**[drawCurve\(\)](#), [drawSymbols\(\)](#)

Definition at line 601 of file qwt_plot_curve.cpp.

References [QwtPlot::canvas\(\)](#), [QwtPlot::canvasMap\(\)](#), [draw\(\)](#), [QwtPlotCanvas::paintCache\(\)](#), [QwtPlotItem::plot\(\)](#), [QwtPlotCanvas::testPaintAttribute\(\)](#), [QwtPlotItem::xAxis\(\)](#), and [QwtPlotItem::yAxis\(\)](#).**6.51.4.36 void QwtPlotCurve::updateLegend (QwtLegend *) const [virtual]**

Update the widget that represents the curve on the legend.

Reimplemented from [QwtPlotItem](#).

Definition at line 1367 of file qwt_plot_curve.cpp.

References [QwtLegend::displayPolicy\(\)](#), [QwtLegend::find\(\)](#), [QwtLegend::identifierMode\(\)](#), [QwtPlotItem::legendItem\(\)](#), [pen\(\)](#), [style\(\)](#), [symbol\(\)](#), [QwtPlotItem::title\(\)](#), and [QwtPlotItem::updateLegend\(\)](#).**6.51.4.37 void QwtPlotCurve::init () [protected]**

Initialize data members.

Definition at line 293 of file qwt_plot_curve.cpp.

References [QwtPlotItem::setItemAttribute\(\)](#), and [QwtPlotItem::setZ\(\)](#).Referenced by [QwtPlotCurve\(\)](#).

6.51.4.38 void QwtPlotCurve::drawCurve (QPainter * *painter*, int *style*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, int *from*, int *to*) const [protected, virtual]

Draw the line part (without symbols) of a curve interval.

Parameters:

painter Painter
style curve style, see QwtPlotCurve::CurveStyle
xMap x map
yMap y map
from index of the first point to be painted
to index of the last point to be painted

See also:

[draw\(\)](#), [drawDots\(\)](#), [drawLines\(\)](#), [drawSteps\(\)](#), [drawSticks\(\)](#)

Definition at line 742 of file qwt_plot_curve.cpp.

References [dataSize\(\)](#), [drawDots\(\)](#), [drawLines\(\)](#), [drawSteps\(\)](#), [drawSticks\(\)](#), and [testCurveAttribute\(\)](#).

Referenced by [draw\(\)](#).

6.51.4.39 void QwtPlotCurve::drawSymbols (QPainter * *painter*, const QwtSymbol & *symbol*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, int *from*, int *to*) const [protected, virtual]

Draw symbols.

Parameters:

painter Painter
symbol Curve symbol
xMap x map
yMap y map
from index of the first point to be painted
to index of the last point to be painted

See also:

[setSymbol\(\)](#), [draw\(\)](#), [drawCurve\(\)](#)

Definition at line 1244 of file qwt_plot_curve.cpp.

References [QwtSymbol::brush\(\)](#), [QwtSymbol::draw\(\)](#), [QwtPainter::metricsMap\(\)](#), [QwtSymbol::pen\(\)](#), [pen\(\)](#), [QwtMetricsMap::screenToLayout\(\)](#), [QwtMetricsMap::screenToLayoutX\(\)](#), [QwtSymbol::size\(\)](#), [symbol\(\)](#), [QwtScaleMap::transform\(\)](#), [x\(\)](#), and [y\(\)](#).

Referenced by [draw\(\)](#).

6.51.4.40 void QwtPlotCurve::drawLines (QPainter * *painter*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, int *from*, int *to*) const [protected]

Draw lines.

If the CurveAttribute Fitted is enabled a QwtCurveFitter tries to interpolate/smooth the curve, before it is painted.

Parameters:

painter Painter

xMap x map

yMap y map

from index of the first point to be painted

to index of the last point to be painted

See also:

[setCurveAttribute\(\)](#), [setCurveFitter\(\)](#), [draw\(\)](#), [drawLines\(\)](#), [drawDots\(\)](#), [drawSteps\(\)](#), [drawSticks\(\)](#)

Definition at line 788 of file qwt_plot_curve.cpp.

References QPainter::drawPolyline(), fillCurve(), QwtScaleMap::transform(), x(), QwtScaleMap::x-Transform(), and y().

Referenced by drawCurve().

6.51.4.41 void QwtPlotCurve::drawSticks (QPainter * *painter*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, int *from*, int *to*) const [protected]

Draw sticks

Parameters:

painter Painter

xMap x map

yMap y map

from index of the first point to be painted

to index of the last point to be painted

See also:

[draw\(\)](#), [drawCurve\(\)](#), [drawDots\(\)](#), [drawLines\(\)](#), [drawSteps\(\)](#)

Definition at line 913 of file qwt_plot_curve.cpp.

References QPainter::drawLine(), QwtScaleMap::transform(), x(), and y().

Referenced by drawCurve().

6.51.4.42 void QwtPlotCurve::drawDots (QPainter * *painter*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, int *from*, int *to*) const [protected]

Draw dots

Parameters:

painter Painter
xMap x map
yMap y map
from index of the first point to be painted
to index of the last point to be painted

See also:

[draw\(\)](#), [drawCurve\(\)](#), [drawSticks\(\)](#), [drawLines\(\)](#), [drawSteps\(\)](#)

Definition at line 943 of file qwt_plot_curve.cpp.

References QwtPainter::drawPoint(), fillCurve(), QwtScaleMap::transform(), x(), and y().

Referenced by drawCurve().

6.51.4.43 void QwtPlotCurve::drawSteps (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, int *from*, int *to*) const [protected]

Draw step function

The direction of the steps depends on Inverted attribute.

Parameters:

painter Painter
xMap x map
yMap y map
from index of the first point to be painted
to index of the last point to be painted

See also:

[CurveAttribute](#), [setCurveAttribute\(\)](#), [draw\(\)](#), [drawCurve\(\)](#), [drawDots\(\)](#), [drawLines\(\)](#), [drawSticks\(\)](#)

Definition at line 1034 of file qwt_plot_curve.cpp.

References QwtPainter::drawPolyline(), fillCurve(), QwtScaleMap::transform(), x(), and y().

Referenced by drawCurve().

6.51.4.44 void QwtPlotCurve::fillCurve (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, QwtPolygon & *pa*) const [protected]

Fill the area between the curve and the baseline with the curve brush

Parameters:

painter Painter
xMap x map
yMap y map
pa Polygon

See also:

[setBrush\(\)](#), [setBaseline\(\)](#), [setCurveType\(\)](#)

Definition at line 1173 of file `qwt_plot_curve.cpp`.

References `closePolyline()`, and `QwtPainter::drawPolygon()`.

Referenced by `drawDots()`, `drawLines()`, and `drawSteps()`.

6.51.4.45 `void QwtPlotCurve::closePolyline (const QwtScaleMap & xMap, const QwtScaleMap & yMap, QwtPolygon & pa) const` [protected]

Complete a polygon to be a closed polygon including the area between the original polygon and the baseline.

Parameters:

xMap X map

yMap Y map

pa Polygon to be completed

Definition at line 1207 of file `qwt_plot_curve.cpp`.

References `QwtScaleMap::transform()`.

Referenced by `fillCurve()`.

6.52 QwtPlotDict Class Reference

Inheritance diagram for QwtPlotDict:



6.52.1 Detailed Description

A dictionary for plot items.

[QwtPlotDict](#) organizes plot items in increasing z-order. If [autoDelete\(\)](#) is enabled, all attached items will be deleted in the destructor of the dictionary.

See also:

[QwtPlotItem::attach\(\)](#), [QwtPlotItem::detach\(\)](#), [QwtPlotItem::z\(\)](#)

Definition at line 42 of file `qwt_plot_dict.h`.

Public Member Functions

- [QwtPlotDict \(\)](#)
- [~QwtPlotDict \(\)](#)

- void [setAutoDelete](#) (bool)
- bool [autoDelete](#) () const
- const [QwtPlotItemList](#) & [itemList](#) () const
- void [detachItems](#) (int rtti=QwtPlotItem::Rtti_PlotItem, bool autoDelete=true)

Friends

- class [QwtPlotItem](#)

6.52.2 Constructor & Destructor Documentation

6.52.2.1 QwtPlotDict::QwtPlotDict () [explicit]

Constructor

Auto deletion is enabled.

See also:

[setAutoDelete](#), [attachItem](#)

Definition at line 94 of file `qwt_plot_dict.cpp`.

6.52.2.2 QwtPlotDict::~~QwtPlotDict ()

Destructor

If `autoDelete` is on, all attached items will be deleted

See also:

[setAutoDelete](#), [autoDelete](#), [attachItem](#)

Definition at line 106 of file `qwt_plot_dict.cpp`.

References [detachItems\(\)](#).

6.52.3 Member Function Documentation

6.52.3.1 void QwtPlotDict::setAutoDelete (bool *autoDelete*)

En/Disable Auto deletion

If Auto deletion is on all attached plot items will be deleted in the destructor of [QwtPlotDict](#). The default value is on.

See also:

[autoDelete](#), [attachItem](#)

Definition at line 120 of file `qwt_plot_dict.cpp`.

6.52.3.2 bool QwtPlotDict::autoDelete () const

Returns:

true if auto deletion is enabled

See also:

[setAutoDelete](#), [attachItem](#)

Definition at line 129 of file qwt_plot_dict.cpp.

Referenced by `QwtPlot::~QwtPlot()`.

6.52.3.3 const QwtPlotItemList & QwtPlotDict::itemList () const

A `QwtPlotItemList` of all attached plot items.

Use caution when iterating these lists, as removing/detaching an item will invalidate the iterator. Instead you can place pointers to objects to be removed in a removal list, and traverse that list later.

Returns:

List of all attached plot items.

Definition at line 186 of file qwt_plot_dict.cpp.

Referenced by `QwtPlotPrintFilter::apply()`, `QwtPlot::drawItems()`, `QwtPlot::insertLegend()`, `QwtPlotPrintFilter::reset()`, and `QwtPlot::updateAxes()`.

6.52.3.4 void QwtPlotDict::detachItems (int rtti = QwtPlotItem::Rtti_PlotItem, bool autoDelete = true)

Detach items from the dictionary

Parameters:

rtti In case of `QwtPlotItem::Rtti_PlotItem` detach all items otherwise only those items of the type `rtti`.

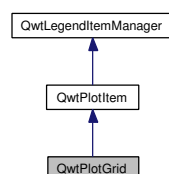
autoDelete If true, delete all detached items

Definition at line 161 of file qwt_plot_dict.cpp.

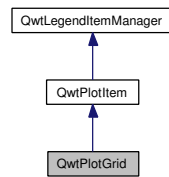
Referenced by `QwtPlot::clear()`, `QwtPlot::~QwtPlot()`, and `~QwtPlotDict()`.

6.53 QwtPlotGrid Class Reference

Inheritance diagram for `QwtPlotGrid`:



Collaboration diagram for QwtPlotGrid:



6.53.1 Detailed Description

A class which draws a coordinate grid.

The [QwtPlotGrid](#) class can be used to draw a coordinate grid. A coordinate grid consists of major and minor vertical and horizontal gridlines. The locations of the gridlines are determined by the X and Y scale divisions which can be assigned with [setXDiv\(\)](#) and [setYDiv\(\)](#). The [draw\(\)](#) member draws the grid within a bounding rectangle.

Definition at line 34 of file `qwt_plot_grid.h`.

Public Member Functions

- [QwtPlotGrid](#) ()
- virtual [~QwtPlotGrid](#) ()
- virtual int [rtti](#) () const
- void [enableX](#) (bool tf)
- bool [xEnabled](#) () const
- void [enableY](#) (bool tf)
- bool [yEnabled](#) () const
- void [enableXMin](#) (bool tf)
- bool [xMinEnabled](#) () const
- void [enableYMin](#) (bool tf)
- bool [yMinEnabled](#) () const
- void [setXDiv](#) (const [QwtScaleDiv](#) &sx)
- const [QwtScaleDiv](#) & [xScaleDiv](#) () const
- void [setYDiv](#) (const [QwtScaleDiv](#) &sy)
- const [QwtScaleDiv](#) & [yScaleDiv](#) () const
- void [setPen](#) (const QPen &p)
- void [setMajPen](#) (const QPen &p)
- const QPen & [majPen](#) () const
- void [setMinPen](#) (const QPen &p)
- const QPen & [minPen](#) () const
- virtual void [draw](#) (QPainter *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const QRect &rect) const
- virtual void [updateScaleDiv](#) (const [QwtScaleDiv](#) &xMap, const [QwtScaleDiv](#) &yMap)

6.53.2 Constructor & Destructor Documentation

6.53.2.1 QwtPlotGrid::QwtPlotGrid () [explicit]

Enables major grid, disables minor grid.

Definition at line 42 of file qwt_plot_grid.cpp.

References [QwtPlotItem::setZ\(\)](#).

6.53.2.2 QwtPlotGrid::~QwtPlotGrid () [virtual]

Destructor.

Definition at line 50 of file qwt_plot_grid.cpp.

6.53.3 Member Function Documentation

6.53.3.1 int QwtPlotGrid::rtti () const [virtual]

Returns:

[QwtPlotItem::Rtti_PlotGrid](#)

Reimplemented from [QwtPlotItem](#).

Definition at line 56 of file qwt_plot_grid.cpp.

6.53.3.2 void QwtPlotGrid::enableX (bool *tf*)

Enable or disable vertical gridlines.

Parameters:

tf Enable (true) or disable

See also:

Minor gridlines can be enabled or disabled with [enableXMin\(\)](#)

Definition at line 68 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.3 bool QwtPlotGrid::xEnabled () const

Returns:

true if vertical gridlines are enabled

See also:

[enableX\(\)](#)

Definition at line 289 of file qwt_plot_grid.cpp.

6.53.3.4 void QwtPlotGrid::enableY (bool *tf*)

Enable or disable horizontal gridlines.

Parameters:

tf Enable (true) or disable

See also:

Minor gridlines can be enabled or disabled with [enableYMin\(\)](#)

Definition at line 82 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.5 bool QwtPlotGrid::yEnabled () const**Returns:**

true if horizontal gridlines are enabled

See also:

[enableY\(\)](#)

Definition at line 307 of file qwt_plot_grid.cpp.

6.53.3.6 void QwtPlotGrid::enableXMin (bool *tf*)

Enable or disable minor vertical gridlines.

Parameters:

tf Enable (true) or disable

See also:

[enableX\(\)](#)

Definition at line 96 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.7 bool QwtPlotGrid::xMinEnabled () const**Returns:**

true if minor vertical gridlines are enabled

See also:

[enableXMin\(\)](#)

Definition at line 298 of file qwt_plot_grid.cpp.

6.53.3.8 void QwtPlotGrid::enableYMin (bool *tf*)

Enable or disable minor horizontal gridlines.

Parameters:

tf Enable (true) or disable

See also:

[enableY\(\)](#)

Definition at line 110 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.9 bool QwtPlotGrid::yMinEnabled () const**Returns:**

true if minor horizontal gridlines are enabled

See also:

[enableYMin\(\)](#)

Definition at line 316 of file qwt_plot_grid.cpp.

6.53.3.10 void QwtPlotGrid::setXDiv (const [QwtScaleDiv](#) & *scaleDiv*)

Assign an x axis scale division.

Parameters:

scaleDiv Scale division

Warning:

[QwtPlotGrid](#) uses implicit sharing (see Qt Manual) for the scale divisions.

Definition at line 125 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

Referenced by [updateScaleDiv\(\)](#).

6.53.3.11 const [QwtScaleDiv](#) & QwtPlotGrid::xScaleDiv () const**Returns:**

the scale division of the x axis

Definition at line 323 of file qwt_plot_grid.cpp.

6.53.3.12 void QwtPlotGrid::setYDiv (const [QwtScaleDiv](#) & sy)

Assign a y axis division.

Parameters:

sy Scale division

Warning:

[QwtPlotGrid](#) uses implicit sharing (see Qt Manual) for the scale divisions.

Definition at line 140 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

Referenced by [updateScaleDiv\(\)](#).

6.53.3.13 const [QwtScaleDiv](#) & QwtPlotGrid::yScaleDiv () const**Returns:**

the scale division of the y axis

Definition at line 329 of file qwt_plot_grid.cpp.

6.53.3.14 void QwtPlotGrid::setPen (const [QPen](#) & p)

Assign a pen for both major and minor gridlines.

Parameters:

p Pen

See also:

[setMajPen\(\)](#), [setMinPen\(\)](#)

Definition at line 154 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.15 void QwtPlotGrid::setMajPen (const [QPen](#) & p)

Assign a pen for the major gridlines.

Parameters:

p Pen

See also:

[majPen\(\)](#), [setMinPen\(\)](#), [setPen\(\)](#)

Definition at line 169 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.16 const QPen & QwtPlotGrid::majPen () const**Returns:**

the pen for the major gridlines

See also:

[setMajPen\(\)](#), [setMinPen\(\)](#), [setPen\(\)](#)

Definition at line 271 of file qwt_plot_grid.cpp.

6.53.3.17 void QwtPlotGrid::setMinPen (const QPen & p)

Assign a pen for the minor gridlines.

Parameters:

p Pen

Definition at line 182 of file qwt_plot_grid.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.53.3.18 const QPen & QwtPlotGrid::minPen () const**Returns:**

the pen for the minor gridlines

See also:

[setMinPen\(\)](#), [setMajPen\(\)](#), [setPen\(\)](#)

Definition at line 280 of file qwt_plot_grid.cpp.

6.53.3.19 void QwtPlotGrid::draw (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, const QRect & *canvasRect*) const [virtual]

Draw the grid.

The grid is drawn into the bounding rectangle such that gridlines begin and end at the rectangle's borders. The X and Y maps are used to map the scale divisions into the drawing region screen.

Parameters:

painter Painter

xMap X axis map

yMap Y axis

canvasRect Contents rect of the plot canvas

Implements [QwtPlotItem](#).

Definition at line 203 of file qwt_plot_grid.cpp.

6.53.3.20 void QwtPlotGrid::updateScaleDiv (const [QwtScaleDiv](#) & *xMap*, const [QwtScaleDiv](#) & *yMap*) [virtual]

Update the item to changes of the axes scale division.

Update the item, when the axes of plot have changed. The default implementation does nothing, but items that depend on the scale division (like [QwtPlotGrid\(\)](#)) have to reimplement [updateScaleDiv\(\)](#)

Parameters:

xScaleDiv Scale division of the x-axis

yScaleDiv Scale division of the y-axis

See also:

[QwtPlot::updateAxes\(\)](#)

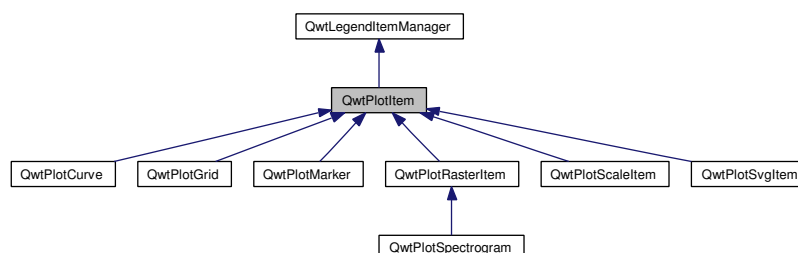
Reimplemented from [QwtPlotItem](#).

Definition at line 334 of file qwt_plot_grid.cpp.

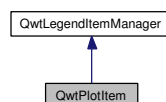
References [setXDiv\(\)](#), and [setYDiv\(\)](#).

6.54 QwtPlotItem Class Reference

Inheritance diagram for QwtPlotItem:



Collaboration diagram for QwtPlotItem:



6.54.1 Detailed Description

Base class for items on the plot canvas.

Definition at line 31 of file qwt_plot_item.h.

Public Types

- enum [RttiValues](#) {

```

    Rtti_PlotItem = 0,
    Rtti_PlotGrid,
    Rtti_PlotScale,
    Rtti_PlotMarker,
    Rtti_PlotCurve,
    Rtti_PlotHistogram,
    Rtti_PlotSpectrogram,
    Rtti_PlotSVG,
    Rtti_PlotUserItem = 1000 }
• enum ItemAttribute {
    Legend = 1,
    AutoScale = 2 }
• enum RenderHint { RenderAntialiased = 1 }

```

Public Member Functions

- [QwtPlotItem](#) (const [QwtText](#) &title=[QwtText](#)())
- virtual [~QwtPlotItem](#) ()
- void [attach](#) ([QwtPlot](#) *plot)
- void [detach](#) ()
- [QwtPlot](#) * [plot](#) () const
- void [setTitle](#) (const [QString](#) &title)
- void [setTitle](#) (const [QwtText](#) &title)
- const [QwtText](#) & [title](#) () const
- virtual int [rtti](#) () const
- void [setItemAttribute](#) ([ItemAttribute](#), bool on=true)
- bool [testItemAttribute](#) ([ItemAttribute](#)) const
- void [setRenderHint](#) ([RenderHint](#), bool on=true)
- bool [testRenderHint](#) ([RenderHint](#)) const
- double [z](#) () const
- void [setZ](#) (double z)
- void [show](#) ()
- void [hide](#) ()
- virtual void [setVisible](#) (bool)
- bool [isVisible](#) () const
- void [setAxis](#) (int xAxis, int yAxis)
- void [setXAxis](#) (int axis)
- int [xAxis](#) () const
- void [setYAxis](#) (int axis)
- int [yAxis](#) () const
- virtual void [itemChanged](#) ()
- virtual void [draw](#) (QPainter *painter, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QRect](#) &canvasRect) const=0
- virtual [QwtDoubleRect](#) [boundingRect](#) () const
- virtual void [updateLegend](#) ([QwtLegend](#) *) const
- virtual void [updateScaleDiv](#) (const [QwtScaleDiv](#) &, const [QwtScaleDiv](#) &)
- virtual [QWidget](#) * [legendItem](#) () const
- [QwtDoubleRect](#) [scaleRect](#) (const [QwtScaleMap](#) &, const [QwtScaleMap](#) &) const

- [QRect paintRect](#) (const [QwtScaleMap](#) &, const [QwtScaleMap](#) &) const
- [QRect transform](#) (const [QwtScaleMap](#) &, const [QwtScaleMap](#) &, const [QwtDoubleRect](#) &) const
- [QwtDoubleRect invTransform](#) (const [QwtScaleMap](#) &, const [QwtScaleMap](#) &, const [QRect](#) &) const

6.54.2 Constructor & Destructor Documentation

6.54.2.1 [QwtPlotItem::QwtPlotItem](#) (const [QwtText](#) & *title* = [QwtText](#) ()) [explicit]

Constructor.

Definition at line 48 of file `qwt_plot_item.cpp`.

References [title\(\)](#).

6.54.2.2 [QwtPlotItem::~QwtPlotItem](#) () [virtual]

Destroy the [QwtPlotItem](#).

Definition at line 55 of file `qwt_plot_item.cpp`.

References [attach\(\)](#).

6.54.3 Member Function Documentation

6.54.3.1 [void QwtPlotItem::attach](#) ([QwtPlot](#) * *plot*)

Attach the item to a plot.

This method will attach a [QwtPlotItem](#) to the [QwtPlot](#) argument. It will first detach the [QwtPlotItem](#) from any plot from a previous call to [attach](#) (if necessary). If a NULL argument is passed, it will detach from any [QwtPlot](#) it was attached to.

See also:

[QwtPlotItem::detach\(\)](#)

Definition at line 71 of file `qwt_plot_item.cpp`.

References [itemChanged\(\)](#), [legendItem\(\)](#), and [plot\(\)](#).

Referenced by [~QwtPlotItem\(\)](#).

6.54.3.2 [void QwtPlotItem::detach](#) () [inline]

This method detaches a [QwtPlotItem](#) from any [QwtPlot](#) it has been associated with.

[detach\(\)](#) is equivalent to calling [attach\(NULL \)](#)

See also:

[attach\(QwtPlot* plot \)](#)

Definition at line 74 of file `qwt_plot_item.h`.

6.54.3.3 [QwtPlot](#) * [QwtPlotItem::plot \(\)](#) const

Return attached plot.

Definition at line 122 of file `qwt_plot_item.cpp`.

Referenced by `attach()`, `QwtPlotCurve::closestPoint()`, `QwtPlotCurve::draw()`, `QwtPlotScaleItem::setScaleDivFromAxis()`, and `QwtPlotScaleItem::setScaleDraw()`.

6.54.3.4 void [QwtPlotItem::setTitle \(const QString & title\)](#)

Set a new title

Parameters:

title Title

See also:

[title\(\)](#)

Definition at line 166 of file `qwt_plot_item.cpp`.

6.54.3.5 void [QwtPlotItem::setTitle \(const \[QwtText\]\(#\) & title\)](#)

Set a new title

Parameters:

title Title

See also:

[title\(\)](#)

Definition at line 177 of file `qwt_plot_item.cpp`.

References `itemChanged()`, and `title()`.

6.54.3.6 const [QwtText](#) & [QwtPlotItem::title \(\)](#) const

Returns:

Title of the item

See also:

[setTitle\(\)](#)

Definition at line 190 of file `qwt_plot_item.cpp`.

Referenced by `QwtPlotItem()`, `setTitle()`, and `QwtPlotCurve::updateLegend()`.

6.54.3.7 int [QwtPlotItem::rtti \(\)](#) const [virtual]

Return rtti for the specific class represented. [QwtPlotItem](#) is simply a virtual interface class, and base classes will implement this method with specific rtti values so a user can differentiate them.

The rtti value is useful for environments, where the runtime type information is disabled and it is not possible to do a `dynamic_cast<...>`.

Returns:

rtti value

See also:

[RttiValues](#)

Reimplemented in [QwtPlotCurve](#), [QwtPlotGrid](#), [QwtPlotMarker](#), [QwtPlotScaleItem](#), [QwtPlotSpectrogram](#), and [QwtPlotSvgItem](#).

Definition at line 116 of file qwt_plot_item.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#), and [QwtPlotPrintFilter::reset\(\)](#).

6.54.3.8 void QwtPlotItem::setItemAttribute ([ItemAttribute](#) attribute, bool on = true)

Toggle an item attribute

Parameters:

attribute Attribute type

on true/false

See also:

[testItemAttribute\(\)](#), [ItemAttribute](#)

Definition at line 203 of file qwt_plot_item.cpp.

References [itemChanged\(\)](#).

Referenced by [QwtPlotCurve::init\(\)](#), and [QwtPlotSpectrogram::QwtPlotSpectrogram\(\)](#).

6.54.3.9 bool QwtPlotItem::testItemAttribute ([ItemAttribute](#) attribute) const

Test an item attribute

Parameters:

ItemAttribute Attribute type

Returns:

true/false

See also:

[setItemAttribute\(\)](#), [ItemAttribute](#)

Definition at line 223 of file qwt_plot_item.cpp.

Referenced by [updateLegend\(\)](#).

6.54.3.10 void QwtPlotItem::setRenderHint ([RenderHint](#) hint, bool on = true)

Toggle an render hint

Parameters:

hint Render hint

on true/false

See also:

[testRenderHint\(\)](#), [RenderHint](#)

Definition at line 238 of file qwt_plot_item.cpp.

References [itemChanged\(\)](#).

6.54.3.11 bool QwtPlotItem::testRenderHint ([RenderHint hint](#)) const

Test a render hint

Parameters:

hint Render hint

Returns:

true/false

See also:

[setRenderHint\(\)](#), [RenderHint](#)

Definition at line 258 of file qwt_plot_item.cpp.

6.54.3.12 double QwtPlotItem::z () const

Plot items are painted in increasing z-order.

Returns:

[setZ\(\)](#), [QwtPlotDict::itemList\(\)](#)

Definition at line 132 of file qwt_plot_item.cpp.

6.54.3.13 void QwtPlotItem::setZ (double z)

Set the z value.

Plot items are painted in increasing z-order.

Parameters:

z Z-value

See also:

[z\(\)](#), [QwtPlotDict::itemList\(\)](#)

Definition at line 145 of file qwt_plot_item.cpp.

References [itemChanged\(\)](#).

Referenced by [QwtPlotCurve::init\(\)](#), [QwtPlotGrid::QwtPlotGrid\(\)](#), [QwtPlotMarker::QwtPlotMarker\(\)](#), [QwtPlotScaleItem::QwtPlotScaleItem\(\)](#), and [QwtPlotSpectrogram::QwtPlotSpectrogram\(\)](#).

6.54.3.14 void QwtPlotItem::show ()

Show the item.

Definition at line 266 of file qwt_plot_item.cpp.

References `setVisible()`.

6.54.3.15 void QwtPlotItem::hide ()

Hide the item.

Definition at line 272 of file qwt_plot_item.cpp.

References `setVisible()`.

6.54.3.16 void QwtPlotItem::setVisible (bool *on*) [virtual]

Show/Hide the item

Parameters:

on Show if true, otherwise hide

See also:

[isVisible\(\)](#), [show\(\)](#), [hide\(\)](#)

Definition at line 283 of file qwt_plot_item.cpp.

References `itemChanged()`.

Referenced by `hide()`, and `show()`.

6.54.3.17 bool QwtPlotItem::isVisible () const**Returns:**

true if visible

See also:

[setVisible\(\)](#), [show\(\)](#), [hide\(\)](#)

Definition at line 296 of file qwt_plot_item.cpp.

6.54.3.18 void QwtPlotItem::setAxis (int *xAxis*, int *yAxis*)

Set X and Y axis

The item will painted according to the coordinates its Axes.

Parameters:

xAxis X Axis

yAxis Y Axis

See also:

[setXAxis\(\)](#), [setYAxis\(\)](#), [xAxis\(\)](#), [yAxis\(\)](#)

Definition at line 328 of file qwt_plot_item.cpp.

References `itemChanged()`.

6.54.3.19 void QwtPlotItem::setXAxis (int *axis*)

Set the X axis

The item will painted according to the coordinates its Axes.

Parameters:

axis X Axis

See also:

[setAxis\(\)](#), [setYAxis\(\)](#), [xAxis\(\)](#)

Definition at line 347 of file qwt_plot_item.cpp.

References `itemChanged()`.

6.54.3.20 int QwtPlotItem::xAxis () const

Return `xAxis`.

Definition at line 374 of file qwt_plot_item.cpp.

Referenced by `QwtPlotCurve::closestPoint()`, `QwtPlotCurve::draw()`, `QwtPlotScaleItem::setScaleDivFromAxis()`, and `QwtPlotScaleItem::setScaleDraw()`.

6.54.3.21 void QwtPlotItem::setYAxis (int *axis*)

Set the Y axis

The item will painted according to the coordinates its Axes.

Parameters:

axis Y Axis

See also:

[setAxis\(\)](#), [setXAxis\(\)](#), [yAxis\(\)](#)

Definition at line 364 of file qwt_plot_item.cpp.

References `itemChanged()`.

6.54.3.22 int QwtPlotItem::yAxis () const

Return `yAxis`.

Definition at line 380 of file qwt_plot_item.cpp.

Referenced by `QwtPlotCurve::closestPoint()`, `QwtPlotCurve::draw()`, `QwtPlotScaleItem::setScaleDivFromAxis()`, and `QwtPlotScaleItem::setScaleDraw()`.

6.54.3.23 void QwtPlotItem::itemChanged () [virtual]

Update the legend and call [QwtPlot::autoRefresh](#) for the parent plot.

See also:

[updateLegend\(\)](#)

Definition at line 307 of file `qwt_plot_item.cpp`.

References [updateLegend\(\)](#).

Referenced by [attach\(\)](#), [QwtPlotGrid::enableX\(\)](#), [QwtPlotGrid::enableXMin\(\)](#), [QwtPlotGrid::enableY\(\)](#), [QwtPlotGrid::enableYMin\(\)](#), [QwtPlotSvgItem::loadData\(\)](#), [QwtPlotSvgItem::loadFile\(\)](#), [QwtPlotScaleItem::setAlignment\(\)](#), [QwtPlotRasterItem::setAlpha\(\)](#), [setAxis\(\)](#), [QwtPlotCurve::setBaseline\(\)](#), [QwtPlotScaleItem::setBorderDistance\(\)](#), [QwtPlotCurve::setBrush\(\)](#), [QwtPlotRasterItem::setCachePolicy\(\)](#), [QwtPlotSpectrogram::setColorMap\(\)](#), [QwtPlotSpectrogram::setConrecAttribute\(\)](#), [QwtPlotSpectrogram::setContourLevels\(\)](#), [QwtPlotCurve::setCurveAttribute\(\)](#), [QwtPlotCurve::setCurveFitter\(\)](#), [QwtPlotCurve::setCurveType\(\)](#), [QwtPlotSpectrogram::setData\(\)](#), [QwtPlotCurve::setData\(\)](#), [QwtPlotSpectrogram::setDefaultContourPen\(\)](#), [QwtPlotSpectrogram::setDisplayMode\(\)](#), [QwtPlotScaleItem::setFont\(\)](#), [setItemAttribute\(\)](#), [QwtPlotMarker::setLabel\(\)](#), [QwtPlotMarker::setLinePen\(\)](#), [QwtPlotMarker::setLineStyle\(\)](#), [QwtPlotGrid::setMajPen\(\)](#), [QwtPlotGrid::setMinPen\(\)](#), [QwtPlotScaleItem::setPalette\(\)](#), [QwtPlotGrid::setPen\(\)](#), [QwtPlotCurve::setPen\(\)](#), [QwtPlotScaleItem::setPosition\(\)](#), [QwtPlotCurve::setRawData\(\)](#), [setRenderHint\(\)](#), [QwtPlotScaleItem::setScaleDivFromAxis\(\)](#), [QwtPlotScaleItem::setScaleDraw\(\)](#), [QwtPlotCurve::setStyle\(\)](#), [QwtPlotMarker::setSymbol\(\)](#), [QwtPlotCurve::setSymbol\(\)](#), [setTitle\(\)](#), [QwtPlotMarker::setValue\(\)](#), [setVisible\(\)](#), [setXAxis\(\)](#), [QwtPlotGrid::setXDiv\(\)](#), [setYAxis\(\)](#), [QwtPlotGrid::setYDiv\(\)](#), and [setZ\(\)](#).

6.54.3.24 virtual void QwtPlotItem::draw (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, const QRect & *canvasRect*) const [pure virtual]

Draw the item.

Parameters:

painter Painter

xMap Maps x-values into pixel coordinates.

yMap Maps y-values into pixel coordinates.

canvasRect Contents rect of the canvas in painter coordinates

Implemented in [QwtPlotCurve](#), [QwtPlotGrid](#), [QwtPlotMarker](#), [QwtPlotRasterItem](#), [QwtPlotScaleItem](#), [QwtPlotSpectrogram](#), and [QwtPlotSvgItem](#).

6.54.3.25 [QwtDoubleRect](#) QwtPlotItem::boundingRect () const [virtual]

Returns:

An invalid bounding rect: `QwtDoubleRect(1.0, 1.0, -2.0, -2.0)`

Reimplemented in [QwtPlotCurve](#), [QwtPlotMarker](#), [QwtPlotSpectrogram](#), and [QwtPlotSvgItem](#).

Definition at line 388 of file `qwt_plot_item.cpp`.

Referenced by [QwtPlotRasterItem::draw\(\)](#), [QwtPlotCurve::maxXValue\(\)](#), [QwtPlotCurve::maxYValue\(\)](#), [QwtPlotCurve::minXValue\(\)](#), and [QwtPlotCurve::minYValue\(\)](#).

6.54.3.26 void QwtPlotItem::updateLegend (QwtLegend * legend) const [virtual]

Update the widget that represents the item on the legend.

[updateLegend\(\)](#) is called from [itemChanged\(\)](#) to adopt the widget representing the item on the legend to its new configuration.

The default implementation is made for [QwtPlotCurve](#) and updates a [QwtLegendItem\(\)](#), but an item could be represented by any type of widget, by overloading [legendItem\(\)](#) and [updateLegend\(\)](#).

See also:

[legendItem\(\)](#), [itemChanged\(\)](#), [QwtLegend\(\)](#)

Implements [QwtLegendItemManager](#).

Reimplemented in [QwtPlotCurve](#).

Definition at line 420 of file `qwt_plot_item.cpp`.

References [QwtLegend::find\(\)](#), [QwtLegend::insert\(\)](#), [QwtLegend::itemMode\(\)](#), [legendItem\(\)](#), and [testItemAttribute\(\)](#).

Referenced by [itemChanged\(\)](#), and [QwtPlotCurve::updateLegend\(\)](#).

6.54.3.27 void QwtPlotItem::updateScaleDiv (const QwtScaleDiv &, const QwtScaleDiv &) [virtual]

Update the item to changes of the axes scale division.

Update the item, when the axes of plot have changed. The default implementation does nothing, but items that depend on the scale division (like [QwtPlotGrid\(\)](#)) have to reimplement [updateScaleDiv\(\)](#)

Parameters:

xScaleDiv Scale division of the x-axis

yScaleDiv Scale division of the y-axis

See also:

[QwtPlot::updateAxes\(\)](#)

Reimplemented in [QwtPlotGrid](#), and [QwtPlotScaleItem](#).

Definition at line 475 of file `qwt_plot_item.cpp`.

6.54.3.28 QWidget * QwtPlotItem::legendItem () const [virtual]

Allocate the widget that represents the item on the legend.

The default implementation is made for [QwtPlotCurve](#) and returns a [QwtLegendItem\(\)](#), but an item could be represented by any type of widget, by overloading [legendItem\(\)](#) and [updateLegend\(\)](#).

Returns:

[QwtLegendItem\(\)](#)

See also:

[updateLegend\(\)](#) [QwtLegend\(\)](#)

Implements [QwtLegendItemManager](#).

Definition at line 403 of file qwt_plot_item.cpp.

Referenced by `attach()`, `updateLegend()`, and `QwtPlotCurve::updateLegend()`.

6.54.3.29 [QwtDoubleRect](#) `QwtPlotItem::scaleRect (const QwtScaleMap & xMap, const QwtScaleMap & yMap) const`

Calculate the bounding scale rect of 2 maps.

Parameters:

xMap X map

yMap X map

Returns:

Bounding rect of the scale maps

Definition at line 488 of file qwt_plot_item.cpp.

References `QwtScaleMap::s1()`, and `QwtScaleMap::sDist()`.

6.54.3.30 `QRect` `QwtPlotItem::paintRect (const QwtScaleMap & xMap, const QwtScaleMap & yMap) const`

Calculate the bounding paint rect of 2 maps.

Parameters:

xMap X map

yMap X map

Returns:

Bounding rect of the scale maps

Definition at line 503 of file qwt_plot_item.cpp.

References `QwtScaleMap::p1()`, and `QwtScaleMap::pDist()`.

Referenced by `QwtPlotRasterItem::draw()`.

6.54.3.31 `QRect` `QwtPlotItem::transform (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtDoubleRect & rect) const`

Transform a rectangle

Parameters:

xMap X map

yMap Y map

rect Rectangle in scale coordinates

Returns:

Rectangle in paint coordinates

See also:

[invTransform\(\)](#)

Definition at line 522 of file qwt_plot_item.cpp.

References [QwtScaleMap::transform\(\)](#).

Referenced by [QwtPlotSvgItem::draw\(\)](#), [QwtPlotSpectrogram::draw\(\)](#), [QwtPlotRasterItem::draw\(\)](#), and [QwtPlotSpectrogram::renderImage\(\)](#).

6.54.3.32 [QwtDoubleRect](#) [QwtPlotItem::invTransform](#) (const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, const [QRect](#) & *rect*) const

Transform a rectangle from paint to scale coordinates

Parameters:

xMap X map

yMap Y map

rect Rectangle in paint coordinates

Returns:

Rectangle in scale coordinates

See also:

[transform\(\)](#)

Definition at line 547 of file qwt_plot_item.cpp.

References [QwtScaleMap::invTransform\(\)](#).

Referenced by [QwtPlotSvgItem::draw\(\)](#), [QwtPlotSpectrogram::draw\(\)](#), and [QwtPlotRasterItem::draw\(\)](#).

6.55 QwtPlotLayout Class Reference

6.55.1 Detailed Description

Layout class for [QwtPlot](#).

Organizes the geometry for the different [QwtPlot](#) components.

Definition at line 22 of file qwt_plot_layout.h.

Public Types

- enum [Options](#) {
 AlignScales = 1,
 IgnoreScrollbars = 2,
 IgnoreFrames = 4,
 IgnoreMargin = 8,
 IgnoreLegend = 16,

```

PrintMargin = 1,
PrintTitle = 2,
PrintLegend = 4,
PrintGrid = 8,
PrintBackground = 16,
PrintFrameWithScales = 32,
PrintAll = ~PrintFrameWithScales }

```

Public Member Functions

- [QwtPlotLayout](#) ()
- virtual [~QwtPlotLayout](#) ()
- void [setMargin](#) (int)
- int [margin](#) () const
- void [setCanvasMargin](#) (int margin, int axis=-1)
- int [canvasMargin](#) (int axis) const
- void [setAlignCanvasToScales](#) (bool)
- bool [alignCanvasToScales](#) () const
- void [setSpacing](#) (int)
- int [spacing](#) () const
- void [setLegendPosition](#) ([QwtPlot::LegendPosition](#) pos, double ratio)
- void [setLegendPosition](#) ([QwtPlot::LegendPosition](#) pos)
- [QwtPlot::LegendPosition](#) [legendPosition](#) () const
- void [setLegendRatio](#) (double ratio)
- double [legendRatio](#) () const
- virtual QSize [minimumSizeHint](#) (const [QwtPlot](#) *) const
- virtual void [activate](#) (const [QwtPlot](#) *, const [QRect](#) &rect, int options=0)
- virtual void [invalidate](#) ()
- const [QRect](#) & [titleRect](#) () const
- const [QRect](#) & [legendRect](#) () const
- const [QRect](#) & [scaleRect](#) (int axis) const
- const [QRect](#) & [canvasRect](#) () const

Protected Member Functions

- [QRect](#) [layoutLegend](#) (int options, const [QRect](#) &) const
- [QRect](#) [alignLegend](#) (const [QRect](#) &canvasRect, const [QRect](#) &legendRect) const
- void [expandLineBreaks](#) (int options, const [QRect](#) &rect, int &dimTitle, int dimAxes[[QwtPlot::axisCnt](#)]) const
- void [alignScales](#) (int options, [QRect](#) &canvasRect, [QRect](#) scaleRect[[QwtPlot::axisCnt](#)]) const

6.55.2 Constructor & Destructor Documentation

6.55.2.1 QwtPlotLayout::QwtPlotLayout () [explicit]

Constructor.

Definition at line 182 of file `qwt_plot_layout.cpp`.

References [invalidate\(\)](#), [setCanvasMargin\(\)](#), and [setLegendPosition\(\)](#).

6.55.2.2 QwtPlotLayout::~~QwtPlotLayout () [virtual]

Destructor.

Definition at line 193 of file qwt_plot_layout.cpp.

6.55.3 Member Function Documentation

6.55.3.1 void QwtPlotLayout::setMargin (int *margin*)

Change the margin of the plot. The margin is the space around all components.

Parameters:

margin new margin

See also:

[margin\(\)](#), [setSpacing\(\)](#), [QwtPlot::setMargin\(\)](#)

Definition at line 206 of file qwt_plot_layout.cpp.

6.55.3.2 int QwtPlotLayout::margin () const

Returns:

margin

See also:

[setMargin\(\)](#), [spacing\(\)](#), [QwtPlot::margin\(\)](#)

Definition at line 217 of file qwt_plot_layout.cpp.

6.55.3.3 void QwtPlotLayout::setCanvasMargin (int *margin*, int *axis* = -1)

Change a margin of the canvas. The margin is the space above/below the scale ticks. A negative margin will be set to -1, excluding the borders of the scales.

Parameters:

margin New margin

axis One of [QwtPlot::Axis](#). Specifies where the position of the margin. -1 means margin at all borders.

See also:

[canvasMargin\(\)](#)

Warning:

The margin will have no effect when `alignCanvasToScales` is true

Definition at line 235 of file qwt_plot_layout.cpp.

Referenced by [QwtPlotLayout\(\)](#).

6.55.3.4 int QwtPlotLayout::canvasMargin (int *axis*) const

Returns:

Margin around the scale tick borders

See also:

[setCanvasMargin\(\)](#)

Definition at line 253 of file qwt_plot_layout.cpp.

Referenced by QwtPlot::canvasMap(), and QwtPlot::print().

6.55.3.5 void QwtPlotLayout::setAlignCanvasToScales (bool *alignCanvasToScales*)

Change the align-canvas-to-axis-scales setting. The canvas may:

- extend beyond the axis scale ends to maximize its size,
- align with the axis scale ends to control its size.

Parameters:

alignCanvasToScales New align-canvas-to-axis-scales setting

See also:

[setCanvasMargin\(\)](#)

Note:

In this context the term 'scale' means the backbone of a scale.

Warning:

In case of alignCanvasToScales == true canvasMargin will have no effect

Definition at line 273 of file qwt_plot_layout.cpp.

6.55.3.6 bool QwtPlotLayout::alignCanvasToScales () const

Return the align-canvas-to-axis-scales setting. The canvas may:

- extend beyond the axis scale ends to maximize its size
- align with the axis scale ends to control its size.

Returns:

align-canvas-to-axis-scales setting

See also:

[setAlignCanvasToScales](#), [setCanvasMargin\(\)](#)

Note:

In this context the term 'scale' means the backbone of a scale.

Definition at line 287 of file qwt_plot_layout.cpp.

6.55.3.7 void QwtPlotLayout::setSpacing (int *spacing*)

Change the spacing of the plot. The spacing is the distance between the plot components.

Parameters:

spacing new spacing

See also:

[setMargin\(\)](#), [spacing\(\)](#)

Definition at line 299 of file qwt_plot_layout.cpp.

6.55.3.8 int QwtPlotLayout::spacing () const

Returns:

spacing

See also:

[margin\(\)](#), [setSpacing\(\)](#)

Definition at line 308 of file qwt_plot_layout.cpp.

6.55.3.9 void QwtPlotLayout::setLegendPosition (QwtPlot::LegendPosition *pos*, double *ratio*)

Specify the position of the legend.

Parameters:

pos The legend's position.

ratio Ratio between legend and the bounding rect of title, canvas and axes. The legend will be shrunk if it would need more space than the given ratio. The ratio is limited to]0.0 .. 1.0]. In case of <= 0.0 it will be reset to the default ratio. The default vertical/horizontal ratio is 0.33/0.5.

See also:

[QwtPlot::setLegendPosition\(\)](#)

Definition at line 326 of file qwt_plot_layout.cpp.

Referenced by [QwtPlotLayout\(\)](#), [setLegendPosition\(\)](#), and [setLegendRatio\(\)](#).

6.55.3.10 void QwtPlotLayout::setLegendPosition (QwtPlot::LegendPosition *pos*)

Specify the position of the legend.

Parameters:

pos The legend's position. Valid values are [QwtPlot::LeftLegend](#), [QwtPlot::RightLegend](#), [QwtPlot::TopLegend](#), [QwtPlot::BottomLegend](#).

See also:

[QwtPlot::setLegendPosition\(\)](#)

Definition at line 363 of file qwt_plot_layout.cpp.

References [setLegendPosition\(\)](#).

6.55.3.11 QwtPlot::LegendPosition QwtPlotLayout::legendPosition () const**Returns:**

Position of the legend

See also:

[setLegendPosition\(\)](#), [QwtPlot::setLegendPosition\(\)](#), [QwtPlot::legendPosition\(\)](#)

Definition at line 373 of file `qwt_plot_layout.cpp`.

Referenced by [setLegendRatio\(\)](#).

6.55.3.12 void QwtPlotLayout::setLegendRatio (double *ratio*)

Specify the relative size of the legend in the plot

Parameters:

ratio Ratio between legend and the bounding rect of title, canvas and axes. The legend will be shrinked if it would need more space than the given ratio. The ratio is limited to]0.0 .. 1.0]. In case of ≤ 0.0 it will be reset to the default ratio. The default vertical/horizontal ratio is 0.33/0.5.

Definition at line 387 of file `qwt_plot_layout.cpp`.

References [legendPosition\(\)](#), and [setLegendPosition\(\)](#).

6.55.3.13 double QwtPlotLayout::legendRatio () const**Returns:**

The relative size of the legend in the plot.

See also:

[setLegendPosition\(\)](#)

Definition at line 396 of file `qwt_plot_layout.cpp`.

6.55.3.14 QSize QwtPlotLayout::minimumSizeHint (const QwtPlot * *plot*) const [virtual]

Return a minimum size hint.

See also:

[QwtPlot::minimumSizeHint\(\)](#)

Definition at line 463 of file `qwt_plot_layout.cpp`.

References [QwtPlot::axisEnabled\(\)](#), [QwtPlot::axisWidget\(\)](#), [QwtPlot::canvas\(\)](#), and [QwtScaleWidget::minimumSizeHint\(\)](#).

6.55.3.15 `void QwtPlotLayout::activate (const QwtPlot * plot, const QRect & plotRect, int options = 0) [virtual]`

Recalculate the geometry of all components.

Parameters:

plot Plot to be layout

plotRect Rect where to place the components

options Options

See also:

[invalidate\(\)](#), [titleRect\(\)](#), [legendRect\(\)](#), [scaleRect\(\)](#), [canvasRect\(\)](#)

Definition at line 1041 of file `qwt_plot_layout.cpp`.

References [alignLegend\(\)](#), [alignScales\(\)](#), [expandLineBreaks\(\)](#), [invalidate\(\)](#), [QwtLegend::isEmpty\(\)](#), [layoutLegend\(\)](#), [QwtPlot::legend\(\)](#), and [scaleRect\(\)](#).

6.55.3.16 `void QwtPlotLayout::invalidate () [virtual]`

Invalidate the geometry of all components.

See also:

[activate\(\)](#)

Definition at line 451 of file `qwt_plot_layout.cpp`.

Referenced by [activate\(\)](#), and [QwtPlotLayout\(\)](#).

6.55.3.17 `const QRect & QwtPlotLayout::titleRect () const`

Returns:

Geometry for the title

See also:

[activate\(\)](#), [invalidate\(\)](#)

Definition at line 406 of file `qwt_plot_layout.cpp`.

6.55.3.18 `const QRect & QwtPlotLayout::legendRect () const`

Returns:

Geometry for the legend

See also:

[activate\(\)](#), [invalidate\(\)](#)

Definition at line 416 of file `qwt_plot_layout.cpp`.

Referenced by [layoutLegend\(\)](#).

6.55.3.19 const QRect & QwtPlotLayout::scaleRect (int *axis*) const**Parameters:**

axis Axis index

Returns:

Geometry for the scale

See also:

[activate\(\)](#), [invalidate\(\)](#)

Definition at line 427 of file qwt_plot_layout.cpp.

Referenced by [activate\(\)](#), [alignScales\(\)](#), and [QwtPlot::print\(\)](#).

6.55.3.20 const QRect & QwtPlotLayout::canvasRect () const**Returns:**

Geometry for the canvas

See also:

[activate\(\)](#), [invalidate\(\)](#)

Definition at line 442 of file qwt_plot_layout.cpp.

Referenced by [QwtPlot::print\(\)](#).

6.55.3.21 QRect QwtPlotLayout::layoutLegend (int *options*, const QRect & *rect*) const
[protected]

Find the geometry for the legend

Parameters:

options Options how to layout the legend

rect Rectangle where to place the legend

Returns:

Geometry for the legend

Definition at line 647 of file qwt_plot_layout.cpp.

References [legendRect\(\)](#).

Referenced by [activate\(\)](#).

6.55.3.22 QRect QwtPlotLayout::alignLegend (const QRect & *canvasRect*, const QRect & *legendRect*) const
[protected]

Align the legend to the canvas

Parameters:

canvasRect Geometry of the canvas
legendRect Maximum geometry for the legend

Returns:

Geometry for the aligned legend

Definition at line 708 of file qwt_plot_layout.cpp.

Referenced by activate().

6.55.3.23 void QwtPlotLayout::expandLineBreaks (int *options*, const QRect & *rect*, int & *dimTitle*, int *dimAxis*[QwtPlot::axisCnt]) const [protected]

Expand all line breaks in text labels, and calculate the height of their widgets in orientation of the text.

Parameters:

options Options how to layout the legend
rect Bounding rect for title, axes and canvas.
dimTitle Expanded height of the title widget
dimAxis Expanded heights of the axis in axis orientation.

Definition at line 743 of file qwt_plot_layout.cpp.

Referenced by activate().

6.55.3.24 void QwtPlotLayout::alignScales (int *options*, QRect & *canvasRect*, QRect *scaleRect*[QwtPlot::axisCnt]) const [protected]

Align the ticks of the axis to the canvas borders using the empty corners.

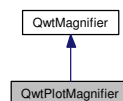
Definition at line 868 of file qwt_plot_layout.cpp.

References scaleRect().

Referenced by activate().

6.56 QwtPlotMagnifier Class Reference

Inheritance diagram for QwtPlotMagnifier:



Collaboration diagram for QwtPlotMagnifier:



6.56.1 Detailed Description

[QwtPlotMagnifier](#) provides zooming, by magnifying in steps.

Using [QwtPlotMagnifier](#) a plot can be zoomed in/out in steps using keys, the mouse wheel or moving a mouse button in vertical direction.

Together with [QwtPlotZoomer](#) and [QwtPlotPanner](#) it is possible to implement individual and powerful navigation of the plot canvas.

See also:

[QwtPlotZoomer](#), [QwtPlotPanner](#), [QwtPlot](#)

Definition at line 30 of file `qwt_plot_magnifier.h`.

Public Member Functions

- [QwtPlotMagnifier](#) ([QwtPlotCanvas](#) *)
- virtual [~QwtPlotMagnifier](#) ()
- void [setAxisEnabled](#) (int axis, bool on)
- bool [isAxisEnabled](#) (int axis) const
- [QwtPlotCanvas](#) * [canvas](#) ()
- const [QwtPlotCanvas](#) * [canvas](#) () const
- [QwtPlot](#) * [plot](#) ()
- const [QwtPlot](#) * [plot](#) () const

Protected Member Functions

- virtual void [rescale](#) (double factor)

6.56.2 Constructor & Destructor Documentation

6.56.2.1 [QwtPlotMagnifier::QwtPlotMagnifier](#) ([QwtPlotCanvas](#) * *canvas*) [explicit]

Constructor

Parameters:

canvas Plot canvas to be magnified

Definition at line 35 of file `qwt_plot_magnifier.cpp`.

6.56.2.2 [QwtPlotMagnifier::~~QwtPlotMagnifier](#) () [virtual]

Destructor.

Definition at line 42 of file `qwt_plot_magnifier.cpp`.

6.56.3 Member Function Documentation

6.56.3.1 void [QwtPlotMagnifier::setAxisEnabled](#) (int *axis*, bool *on*)

En/Disable an axis.

Axes that are enabled will be synchronized to the result of panning. All other axes will remain unchanged.

Parameters:

axis Axis, see [QwtPlot::Axis](#)

on On/Off

See also:

[isAxisEnabled](#)

Definition at line 58 of file qwt_plot_magnifier.cpp.

6.56.3.2 bool QwtPlotMagnifier::isAxisEnabled (int *axis*) const

Test if an axis is enabled

Parameters:

axis Axis, see [QwtPlot::Axis](#)

Returns:

True, if the axis is enabled

See also:

[setAxisEnabled](#)

Definition at line 72 of file qwt_plot_magnifier.cpp.

Referenced by [rescale\(\)](#).

6.56.3.3 QwtPlotCanvas * QwtPlotMagnifier::canvas ()

Return observed plot canvas.

Definition at line 81 of file qwt_plot_magnifier.cpp.

Referenced by [plot\(\)](#).

6.56.3.4 const QwtPlotCanvas * QwtPlotMagnifier::canvas () const

Return Observed plot canvas.

Definition at line 91 of file qwt_plot_magnifier.cpp.

6.56.3.5 QwtPlot * QwtPlotMagnifier::plot ()

Return plot widget, containing the observed plot canvas.

Definition at line 97 of file qwt_plot_magnifier.cpp.

References [canvas\(\)](#).

Referenced by [rescale\(\)](#).

6.56.3.6 const QwtPlot * QwtPlotMagnifier::plot () const

Return plot widget, containing the observed plot canvas.

Definition at line 111 of file qwt_plot_magnifier.cpp.

6.56.3.7 void QwtPlotMagnifier::rescale (double *factor*) [protected, virtual]

Zoom in/out the axes scales

Parameters:

factor A value < 1.0 zooms in, a value > 1.0 zooms out.

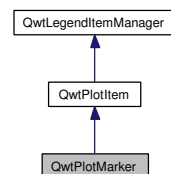
Implements [QwtMagnifier](#).

Definition at line 120 of file qwt_plot_magnifier.cpp.

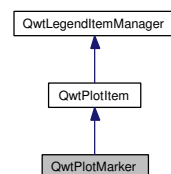
References [QwtPlot::autoReplot\(\)](#), [QwtPlot::axisScaleDiv\(\)](#), [isAxisEnabled\(\)](#), [QwtScaleDiv::lBound\(\)](#), [plot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlot::setAutoReplot\(\)](#), and [QwtPlot::setAxisScale\(\)](#).

6.57 QwtPlotMarker Class Reference

Inheritance diagram for QwtPlotMarker:



Collaboration diagram for QwtPlotMarker:

**6.57.1 Detailed Description**

A class for drawing markers.

A marker can be a horizontal line, a vertical line, a symbol, a label or any combination of them, which can be drawn around a center point inside a bounding rectangle.

The [QwtPlotMarker::setSymbol\(\)](#) member assigns a symbol to the marker. The symbol is drawn at the specified point.

With [QwtPlotMarker::setLabel\(\)](#), a label can be assigned to the marker. The [QwtPlotMarker::setLabelAlignment\(\)](#) member specifies where the label is drawn. All the Align*-constants in Qt::AlignmentFlags (see Qt documentation) are valid. The interpretation of the alignment depends on the marker's line style. The alignment refers to the center point of the marker, which means, for example, that the label would be printed left above the center point if the alignment was set to AlignLeft|AlignTop.

Definition at line 45 of file qwt_plot_marker.h.

Public Types

- enum [LineStyle](#) {
 NoLine,
 HLine,
 VLine,
 Cross }

Public Member Functions

- [QwtPlotMarker](#) ()
- virtual [~QwtPlotMarker](#) ()
- virtual int [rtti](#) () const
- double [xValue](#) () const
- double [yValue](#) () const
- [QwtDoublePoint](#) [value](#) () const
- void [setXValue](#) (double)
- void [setYValue](#) (double)
- void [setValue](#) (double, double)
- void [setValue](#) (const [QwtDoublePoint](#) &)
- void [setLineStyle](#) ([LineStyle](#) st)
- [LineStyle](#) [lineStyle](#) () const
- void [setLinePen](#) (const [QPen](#) &p)
- const [QPen](#) & [linePen](#) () const
- void [setSymbol](#) (const [QwtSymbol](#) &s)
- const [QwtSymbol](#) & [symbol](#) () const
- void [setLabel](#) (const [QwtText](#) &)
- [QwtText](#) [label](#) () const
- void [setLabelAlignment](#) (Qt::Alignment)
- Qt::Alignment [labelAlignment](#) () const
- virtual void [draw](#) (QPainter *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QRect](#) &) const
- virtual [QwtDoubleRect](#) [boundingRect](#) () const

6.57.2 Member Enumeration Documentation

6.57.2.1 enum [QwtPlotMarker::LineStyle](#)

Line styles.

See also:

[QwtPlotMarker::setLineStyle](#), [QwtPlotMarker::lineStyle](#)

Definition at line 53 of file `qwt_plot_marker.h`.

6.57.3 Constructor & Destructor Documentation

6.57.3.1 QwtPlotMarker::QwtPlotMarker () [explicit]

Sets alignment to Qt::AlignCenter, and style to NoLine.

Definition at line 54 of file qwt_plot_marker.cpp.

References QwtPlotItem::setZ().

6.57.3.2 QwtPlotMarker::~QwtPlotMarker () [virtual]

Destructor.

Definition at line 62 of file qwt_plot_marker.cpp.

6.57.4 Member Function Documentation

6.57.4.1 int QwtPlotMarker::rtti () const [virtual]

Returns:

QwtPlotItem::Rtti_PlotMarker

Reimplemented from [QwtPlotItem](#).

Definition at line 68 of file qwt_plot_marker.cpp.

6.57.4.2 double QwtPlotMarker::xValue () const

Return x Value.

Definition at line 80 of file qwt_plot_marker.cpp.

6.57.4.3 double QwtPlotMarker::yValue () const

Return y Value.

Definition at line 86 of file qwt_plot_marker.cpp.

6.57.4.4 [QwtDoublePoint](#) QwtPlotMarker::value () const

Return Value.

Definition at line 74 of file qwt_plot_marker.cpp.

6.57.4.5 void QwtPlotMarker::setXValue (double)

Set X Value.

Definition at line 109 of file qwt_plot_marker.cpp.

References setValue().

6.57.4.6 void QwtPlotMarker::setYValue (double)

Set Y Value.

Definition at line 115 of file qwt_plot_marker.cpp.

References `setValue()`.

6.57.4.7 void QwtPlotMarker::setValue (double, double)

Set Value.

Definition at line 98 of file qwt_plot_marker.cpp.

References `QwtPlotItem::itemChanged()`.

Referenced by `setValue()`, `setXValue()`, and `setYValue()`.

6.57.4.8 void QwtPlotMarker::setValue (const QwtDoublePoint &)

Set Value.

Definition at line 92 of file qwt_plot_marker.cpp.

References `setValue()`.

6.57.4.9 void QwtPlotMarker::setLineStyle (QwtPlotMarker::LineStyle st)

Set the line style.

Parameters:

st Line style. Can be one of `QwtPlotMarker::NoLine`, `HLine`, `VLine` or `Cross`

See also:

[lineStyle\(\)](#)

Definition at line 236 of file qwt_plot_marker.cpp.

References `QwtPlotItem::itemChanged()`.

6.57.4.10 QwtPlotMarker::LineStyle QwtPlotMarker::lineStyle () const

Returns:

the line style

See also:

For a description of line styles, see [QwtPlotMarker::setLineStyle\(\)](#)

Definition at line 249 of file qwt_plot_marker.cpp.

6.57.4.11 void QwtPlotMarker::setLinePen (const QPen & p)

Specify a pen for the line.

Parameters:

p New pen

See also:

[linePen\(\)](#)

Definition at line 340 of file qwt_plot_marker.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

Referenced by [QwtPlotPrintFilter::apply\(\)](#).

6.57.4.12 const QPen & QwtPlotMarker::linePen () const

Returns:

the line pen

See also:

[setLinePen\(\)](#)

Definition at line 353 of file qwt_plot_marker.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#).

6.57.4.13 void QwtPlotMarker::setSymbol (const QwtSymbol & s)

Assign a symbol.

Parameters:

s New symbol

See also:

[symbol\(\)](#)

Definition at line 259 of file qwt_plot_marker.cpp.

References [QwtSymbol::clone\(\)](#), and [QwtPlotItem::itemChanged\(\)](#).

Referenced by [QwtPlotPrintFilter::apply\(\)](#).

6.57.4.14 const QwtSymbol & QwtPlotMarker::symbol () const

Returns:

the symbol

See also:

[setSymbol\(\)](#), [QwtSymbol](#)

Definition at line 270 of file qwt_plot_marker.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#).

6.57.4.15 void QwtPlotMarker::setLabel (const [QwtText](#) & *label*)

Set the label.

Parameters:

label label text

See also:

[label\(\)](#)

Definition at line 280 of file qwt_plot_marker.cpp.

References [QwtPlotItem::itemChanged\(\)](#), and [label\(\)](#).

Referenced by [QwtPlotPrintFilter::apply\(\)](#).

6.57.4.16 [QwtText](#) QwtPlotMarker::label () const**Returns:**

the label

See also:

[setLabel\(\)](#)

Definition at line 293 of file qwt_plot_marker.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#), and [setLabel\(\)](#).

6.57.4.17 void QwtPlotMarker::setLabelAlignment (Qt::Alignment *align*)

Set the alignment of the label.

The alignment determines where the label is drawn relative to the marker's position.

Parameters:

align Alignment. A combination of [AlignTop](#), [AlignBottom](#), [AlignLeft](#), [AlignRight](#), [AlignCenter](#), [AlignHCenter](#), [AlignVCenter](#).

See also:

[labelAlignment\(\)](#)

Definition at line 312 of file qwt_plot_marker.cpp.

6.57.4.18 Qt::Alignment QwtPlotMarker::labelAlignment () const**Returns:**

the label alignment

See also:

[setLabelAlignment\(\)](#)

Definition at line 329 of file qwt_plot_marker.cpp.

6.57.4.19 `void QwtPlotMarker::draw (QPainter *p, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRect & r) const` [virtual]

Draw the marker.

Parameters:

p Painter
xMap x Scale Map
yMap y Scale Map
r Bounding rect, where to paint

Implements [QwtPlotItem](#).

Definition at line 127 of file qwt_plot_marker.cpp.

References [QwtPainter::drawLine\(\)](#), [QwtPainter::metricsMap\(\)](#), [QwtMetricsMap::screenToLayoutX\(\)](#), [QwtMetricsMap::screenToLayoutY\(\)](#), and [QwtScaleMap::transform\(\)](#).

6.57.4.20 `QwtDoubleRect QwtPlotMarker::boundingRect () const` [virtual]

Returns:

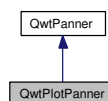
An invalid bounding rect: `QwtDoubleRect(1.0, 1.0, -2.0, -2.0)`

Reimplemented from [QwtPlotItem](#).

Definition at line 358 of file qwt_plot_marker.cpp.

6.58 QwtPlotPanner Class Reference

Inheritance diagram for QwtPlotPanner:



Collaboration diagram for QwtPlotPanner:



6.58.1 Detailed Description

[QwtPlotPanner](#) provides panning of a plot canvas.

[QwtPlotPanner](#) is a panner for a [QwtPlotCanvas](#), that adjusts the scales of the axes after dropping the canvas on its new position.

Together with [QwtPlotZoomer](#) and [QwtPlotMagnifier](#) powerful ways of navigating on a [QwtPlot](#) widget can be implemented easily.

Note:

The axes are not updated, while dragging the canvas

See also:

[QwtPlotZoomer](#), [QwtPlotMagnifier](#)

Definition at line 32 of file qwt_plot_panner.h.

Public Member Functions

- [QwtPlotPanner](#) ([QwtPlotCanvas](#) *)
- virtual [~QwtPlotPanner](#) ()
- [QwtPlotCanvas](#) * [canvas](#) ()
- const [QwtPlotCanvas](#) * [canvas](#) () const
- [QwtPlot](#) * [plot](#) ()
- const [QwtPlot](#) * [plot](#) () const
- void [setAxisEnabled](#) (int axis, bool on)
- bool [isAxisEnabled](#) (int axis) const

Protected Slots

- virtual void [moveCanvas](#) (int dx, int dy)

6.58.2 Constructor & Destructor Documentation**6.58.2.1 QwtPlotPanner::QwtPlotPanner ([QwtPlotCanvas](#) * *canvas*)** [explicit]

Create a plot panner.

The panner is enabled for all axes

Parameters:

canvas Plot canvas to pan, also the parent object

See also:

[setAxisEnabled](#)

Definition at line 38 of file qwt_plot_panner.cpp.

References [moveCanvas\(\)](#), and [QwtPanner::panned\(\)](#).

6.58.2.2 QwtPlotPanner::~QwtPlotPanner () [virtual]

Destructor.

Definition at line 48 of file qwt_plot_panner.cpp.

6.58.3 Member Function Documentation

6.58.3.1 [QwtPlotCanvas](#) * [QwtPlotPanner::canvas](#) ()

Return observed plot canvas.

Definition at line 87 of file `qwt_plot_panner.cpp`.

Referenced by `plot()`.

6.58.3.2 `const QwtPlotCanvas * QwtPlotPanner::canvas () const`

Return Observed plot canvas.

Definition at line 97 of file `qwt_plot_panner.cpp`.

6.58.3.3 [QwtPlot](#) * [QwtPlotPanner::plot](#) ()

Return plot widget, containing the observed plot canvas.

Definition at line 103 of file `qwt_plot_panner.cpp`.

References `canvas()`.

Referenced by `moveCanvas()`.

6.58.3.4 `const QwtPlot * QwtPlotPanner::plot () const`

Return plot widget, containing the observed plot canvas.

Definition at line 117 of file `qwt_plot_panner.cpp`.

6.58.3.5 `void QwtPlotPanner::setAxisEnabled (int axis, bool on)`

En/Disable an axis.

Axes that are enabled will be synchronized to the result of panning. All other axes will remain unchanged.

Parameters:

axis Axis, see [QwtPlot::Axis](#)

on On/Off

See also:

[isAxisEnabled](#), [moveCanvas](#)

Definition at line 64 of file `qwt_plot_panner.cpp`.

6.58.3.6 `bool QwtPlotPanner::isAxisEnabled (int axis) const`

Test if an axis is enabled

Parameters:

axis Axis, see [QwtPlot::Axis](#)

Returns:

True, if the axis is enabled

See also:

[setAxisEnabled](#), [moveCanvas](#)

Definition at line 78 of file qwt_plot_panner.cpp.

6.58.3.7 void QwtPlotPanner::moveCanvas (int *dx*, int *dy*) [protected, virtual, slot]

Adjust the enabled axes according to dx/dy

Parameters:

dx Pixel offset in x direction

dy Pixel offset in y direction

See also:

[QwtPanner::panned\(\)](#)

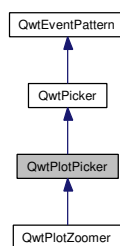
Definition at line 130 of file qwt_plot_panner.cpp.

References [QwtPlot::autoReplot\(\)](#), [QwtPlot::axisScaleDiv\(\)](#), [QwtPlot::canvasMap\(\)](#), [QwtScaleDiv::hBound\(\)](#), [QwtScaleDiv::lBound\(\)](#), [plot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlot::setAutoReplot\(\)](#), [QwtPlot::setAxisScale\(\)](#), and [QwtScaleMap::transform\(\)](#).

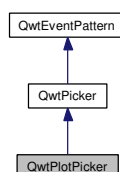
Referenced by [QwtPlotPanner\(\)](#).

6.59 QwtPlotPicker Class Reference

Inheritance diagram for QwtPlotPicker:



Collaboration diagram for QwtPlotPicker:



6.59.1 Detailed Description

[QwtPlotPicker](#) provides selections on a plot canvas.

[QwtPlotPicker](#) is a [QwtPicker](#) tailored for selections on a plot canvas. It is set to a x-Axis and y-Axis and translates all pixel coordinates into this coordinate system.

Definition at line 29 of file `qwt_plot_picker.h`.

Signals

- void [selected](#) (const [QwtDoublePoint](#) &pos)
- void [selected](#) (const [QwtDoubleRect](#) &rect)
- void [selected](#) (const [QwtArray](#)< [QwtDoublePoint](#) > &pa)
- void [appended](#) (const [QwtDoublePoint](#) &pos)
- void [moved](#) (const [QwtDoublePoint](#) &pos)

Public Member Functions

- [QwtPlotPicker](#) ([QwtPlotCanvas](#) *)
- [QwtPlotPicker](#) (int xAxis, int yAxis, [QwtPlotCanvas](#) *)
- [QwtPlotPicker](#) (int xAxis, int yAxis, int selectionFlags, [RubberBand](#) rubberBand, [DisplayMode](#) trackerMode, [QwtPlotCanvas](#) *)
- virtual void [setAxis](#) (int xAxis, int yAxis)
- int [xAxis](#) () const
- int [yAxis](#) () const
- [QwtPlot](#) * [plot](#) ()
- const [QwtPlot](#) * [plot](#) () const
- [QwtPlotCanvas](#) * [canvas](#) ()
- const [QwtPlotCanvas](#) * [canvas](#) () const

Protected Member Functions

- [QwtDoubleRect](#) [scaleRect](#) () const
- [QwtDoubleRect](#) [invTransform](#) (const [QRect](#) &) const
- [QRect](#) [transform](#) (const [QwtDoubleRect](#) &) const
- [QwtDoublePoint](#) [invTransform](#) (const [QPoint](#) &) const
- [QPoint](#) [transform](#) (const [QwtDoublePoint](#) &) const
- virtual [QwtText](#) [trackerText](#) (const [QPoint](#) &) const
- virtual [QwtText](#) [trackerText](#) (const [QwtDoublePoint](#) &) const
- virtual void [move](#) (const [QPoint](#) &)
- virtual void [append](#) (const [QPoint](#) &)
- virtual bool [end](#) (bool ok=true)

6.59.2 Constructor & Destructor Documentation

6.59.2.1 [QwtPlotPicker::QwtPlotPicker](#) ([QwtPlotCanvas](#) * canvas) [explicit]

Create a plot picker.

The picker is set to those x- and y-axis of the plot that are enabled. If both or no x-axis are enabled, the picker is set to `QwtPlot::xBottom`. If both or no y-axis are enabled, it is set to `QwtPlot::yLeft`.

Parameters:

canvas Plot canvas to observe, also the parent object

See also:

[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlotPicker::scaleRect\(\)](#)

Definition at line 32 of file `qwt_plot_picker.cpp`.

References [QwtPlot::axisEnabled\(\)](#), [canvas\(\)](#), [plot\(\)](#), [setAxis\(\)](#), [xAxis\(\)](#), and [yAxis\(\)](#).

6.59.2.2 QwtPlotPicker::QwtPlotPicker (int *xAxis*, int *yAxis*, [QwtPlotCanvas](#) * *canvas*) [explicit]

Create a plot picker

Parameters:

xAxis Set the x axis of the picker

yAxis Set the y axis of the picker

canvas Plot canvas to observe, also the parent object

See also:

[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlotPicker::scaleRect\(\)](#)

Definition at line 70 of file `qwt_plot_picker.cpp`.

6.59.2.3 QwtPlotPicker::QwtPlotPicker (int *xAxis*, int *yAxis*, int *selectionFlags*, [RubberBand](#) *rubberBand*, [DisplayMode](#) *trackerMode*, [QwtPlotCanvas](#) * *canvas*) [explicit]

Create a plot picker

Parameters:

xAxis X axis of the picker

yAxis Y axis of the picker

selectionFlags Or'd value of [SelectionType](#), [RectSelectionType](#) and [SelectionMode](#)

rubberBand Rubberband style

trackerMode Tracker mode

canvas Plot canvas to observe, also the parent object

See also:

[QwtPicker](#), [QwtPicker::setSelectionFlags\(\)](#), [QwtPicker::setRubberBand\(\)](#), [QwtPicker::setTrackerMode\(\)](#), [QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlotPicker::scaleRect\(\)](#)

Definition at line 93 of file `qwt_plot_picker.cpp`.

6.59.3 Member Function Documentation

6.59.3.1 void QwtPlotPicker::setAxis (int xAxis, int yAxis) [virtual]

Set the x and y axes of the picker

Parameters:

xAxis X axis

yAxis Y axis

Reimplemented in [QwtPlotZoomer](#).

Definition at line 169 of file qwt_plot_picker.cpp.

References [plot\(\)](#).

Referenced by [QwtPlotPicker\(\)](#), and [QwtPlotZoomer::setAxis\(\)](#).

6.59.3.2 int QwtPlotPicker::xAxis () const

Return x axis.

Definition at line 183 of file qwt_plot_picker.cpp.

Referenced by [QwtPlotPicker\(\)](#), [QwtPlotZoomer::rescale\(\)](#), [scaleRect\(\)](#), and [QwtPlotZoomer::setAxis\(\)](#).

6.59.3.3 int QwtPlotPicker::yAxis () const

Return y axis.

Definition at line 189 of file qwt_plot_picker.cpp.

Referenced by [QwtPlotPicker\(\)](#), [QwtPlotZoomer::rescale\(\)](#), [scaleRect\(\)](#), and [QwtPlotZoomer::setAxis\(\)](#).

6.59.3.4 [QwtPlot](#) * QwtPlotPicker::plot ()

Return plot widget, containing the observed plot canvas.

Definition at line 119 of file qwt_plot_picker.cpp.

References [canvas\(\)](#).

Referenced by [QwtPlotZoomer::end\(\)](#), [end\(\)](#), [invTransform\(\)](#), [QwtPlotPicker\(\)](#), [QwtPlotZoomer::rescale\(\)](#), [scaleRect\(\)](#), [setAxis\(\)](#), [QwtPlotZoomer::setZoomBase\(\)](#), and [transform\(\)](#).

6.59.3.5 const [QwtPlot](#) * QwtPlotPicker::plot () const

Return plot widget, containing the observed plot canvas.

Definition at line 133 of file qwt_plot_picker.cpp.

6.59.3.6 [QwtPlotCanvas](#) * QwtPlotPicker::canvas ()

Return observed plot canvas.

Definition at line 103 of file qwt_plot_picker.cpp.

References [QwtPicker::parentWidget\(\)](#).

Referenced by [plot\(\)](#), [QwtPlotPicker\(\)](#), and [QwtPlotZoomer::QwtPlotZoomer\(\)](#).

6.59.3.7 `const QwtPlotCanvas * QwtPlotPicker::canvas () const`

Return Observed plot canvas.

Definition at line 113 of file qwt_plot_picker.cpp.

6.59.3.8 `void QwtPlotPicker::selected (const QwtDoublePoint & pos) [signal]`

A signal emitted in case of [selectionFlags\(\)](#) & PointSelection.

Parameters:

pos Selected point

Referenced by end().

6.59.3.9 `void QwtPlotPicker::selected (const QwtDoubleRect & rect) [signal]`

A signal emitted in case of [selectionFlags\(\)](#) & RectSelection.

Parameters:

rect Selected rectangle

6.59.3.10 `void QwtPlotPicker::selected (const QwtArray< QwtDoublePoint > & pa) [signal]`

A signal emitting the selected points, at the end of a selection.

Parameters:

pa Selected points

6.59.3.11 `void QwtPlotPicker::appended (const QwtDoublePoint & pos) [signal]`

A signal emitted when a point has been appended to the selection

Parameters:

pos Position of the appended point.

See also:

[append\(\)](#). [moved\(\)](#)

Referenced by append().

6.59.3.12 `void QwtPlotPicker::moved (const QwtDoublePoint & pos) [signal]`

A signal emitted whenever the last appended point of the selection has been moved.

Parameters:

pos Position of the moved last point of the selection.

See also:

[move\(\)](#), [appended\(\)](#)

Referenced by [move\(\)](#).

6.59.3.13 **QwtDoubleRect** QwtPlotPicker::scaleRect () const [protected]

Return normalized bounding rect of the axes

See also:

[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#).

Definition at line 143 of file `qwt_plot_picker.cpp`.

References [QwtPlot::axisScaleDiv\(\)](#), [QwtScaleDiv::lBound\(\)](#), [plot\(\)](#), [QwtScaleDiv::range\(\)](#), [xAxis\(\)](#), and [yAxis\(\)](#).

Referenced by [QwtPlotZoomer::rescale\(\)](#), [QwtPlotZoomer::setAxis\(\)](#), and [QwtPlotZoomer::setZoomBase\(\)](#).

6.59.3.14 **QwtDoubleRect** QwtPlotPicker::invTransform (const QRect & rect) const [protected]

Translate a rectangle from pixel into plot coordinates

Returns:

Rectangle in plot coordinates

See also:

[QwtPlotPicker::transform\(\)](#)

Definition at line 332 of file `qwt_plot_picker.cpp`.

References [QwtPlot::canvasMap\(\)](#), [QwtScaleMap::invTransform\(\)](#), and [plot\(\)](#).

Referenced by [append\(\)](#), [QwtPlotZoomer::end\(\)](#), [end\(\)](#), [move\(\)](#), and [trackerText\(\)](#).

6.59.3.15 **QRect** QwtPlotPicker::transform (const QwtDoubleRect & rect) const [protected]

Translate a rectangle from plot into pixel coordinates

Returns:

Rectangle in pixel coordinates

See also:

[QwtPlotPicker::invTransform\(\)](#)

Definition at line 351 of file `qwt_plot_picker.cpp`.

References [QwtPlot::canvasMap\(\)](#), [plot\(\)](#), and [QwtScaleMap::transform\(\)](#).

6.59.3.16 [QwtDoublePoint](#) `QwtPlotPicker::invTransform (const QPoint & pos) const` [protected]

Translate a point from pixel into plot coordinates

Returns:

Point in plot coordinates

See also:

[QwtPlotPicker::transform\(\)](#)

Definition at line 369 of file `qwt_plot_picker.cpp`.

References `QwtPlot::canvasMap()`, `QwtScaleMap::invTransform()`, and `plot()`.

6.59.3.17 `QPoint QwtPlotPicker::transform (const QwtDoublePoint & pos) const` [protected]

Translate a point from plot into pixel coordinates

Returns:

Point in pixel coordinates

See also:

[QwtPlotPicker::invTransform\(\)](#)

Definition at line 385 of file `qwt_plot_picker.cpp`.

References `QwtPlot::canvasMap()`, `plot()`, and `QwtScaleMap::transform()`.

6.59.3.18 [QwtText](#) `QwtPlotPicker::trackerText (const QPoint & pos) const` [protected, virtual]

Translate a pixel position into a position string

Parameters:

pos Position in pixel coordinates

Returns:

Position string

Reimplemented from [QwtPicker](#).

Definition at line 200 of file `qwt_plot_picker.cpp`.

References `invTransform()`.

6.59.3.19 [QwtText](#) `QwtPlotPicker::trackerText (const QwtDoublePoint & pos) const` [protected, virtual]

Translate a position into a position string.

In case of HLineRubberBand the label is the value of the y position, in case of VLineRubberBand the value of the x position. Otherwise the label contains x and y position separated by a ','.

The format for the double to string conversion is "%.4f".

Parameters:

pos Position

Returns:

Position string

Definition at line 217 of file qwt_plot_picker.cpp.

References QwtPicker::rubberBand().

6.59.3.20 void QwtPlotPicker::move (const QPoint & pos) [protected, virtual]

Move the last point of the selection

Parameters:

pos New position

See also:

[isActive](#), [begin\(\)](#), [end\(\)](#), [append\(\)](#)

Note:

The [moved\(const QPoint &\)](#), [moved\(const QDoublePoint &\)](#) signals are emitted.

Reimplemented from [QwtPicker](#).

Definition at line 259 of file qwt_plot_picker.cpp.

References [invTransform\(\)](#), [QwtPicker::move\(\)](#), and [moved\(\)](#).

6.59.3.21 void QwtPlotPicker::append (const QPoint & pos) [protected, virtual]

Append a point to the selection and update rubberband and tracker.

Parameters:

pos Additional point

See also:

[isActive](#), [begin\(\)](#), [end\(\)](#), [move\(\)](#), [appended\(\)](#)

Note:

The [appended\(const QPoint &\)](#), [appended\(const QDoublePoint &\)](#) signals are emitted.

Reimplemented from [QwtPicker](#).

Definition at line 244 of file qwt_plot_picker.cpp.

References [QwtPicker::append\(\)](#), [appended\(\)](#), and [invTransform\(\)](#).

6.59.3.22 `bool QwtPlotPicker::end (bool ok = true) [protected, virtual]`

Close a selection setting the state to inactive.

Parameters:

ok If true, complete the selection and emit selected signals otherwise discard the selection.

Returns:

true if the selection is accepted, false otherwise

Reimplemented from [QwtPicker](#).

Reimplemented in [QwtPlotZoomer](#).

Definition at line 273 of file `qwt_plot_picker.cpp`.

References [QwtPicker::end\(\)](#), [invTransform\(\)](#), [plot\(\)](#), [selected\(\)](#), [QwtPicker::selection\(\)](#), and [QwtPicker::selectionFlags\(\)](#).

Referenced by [QwtPlotZoomer::end\(\)](#).

6.60 QwtPlotPrintFilter Class Reference**6.60.1 Detailed Description**

A base class for plot print filters.

A print filter can be used to customize [QwtPlot::print\(\)](#).

Deprecated

In Qwt 5.0 the design of [QwtPlot](#) allows/recommends writing individual [QwtPlotItems](#), that are not known to [QwtPlotPrintFilter](#). So this concept is outdated and [QwtPlotPrintFilter](#) will be removed/replaced in Qwt 6.x.

Definition at line 30 of file `qwt_plot_printfilter.h`.

Public Types

- enum [Options](#) {
 - AlignScales** = 1,
 - IgnoreScrollbars** = 2,
 - IgnoreFrames** = 4,
 - IgnoreMargin** = 8,
 - IgnoreLegend** = 16,
 - PrintMargin** = 1,
 - PrintTitle** = 2,
 - PrintLegend** = 4,
 - PrintGrid** = 8,
 - PrintBackground** = 16,
 - PrintFrameWithScales** = 32,
 - PrintAll** = ~PrintFrameWithScales }

- enum [Item](#) {
 Title,
 Legend,
 Curve,
 CurveSymbol,
 Marker,
 MarkerSymbol,
 MajorGrid,
 MinorGrid,
 CanvasBackground,
 AxisScale,
 AxisTitle,
 WidgetBackground }

Public Member Functions

- [QwtPlotPrintFilter](#) ()
- virtual [~QwtPlotPrintFilter](#) ()
- virtual QColor [color](#) (const QColor &, [Item](#) item) const
- virtual QFont [font](#) (const QFont &, [Item](#) item) const
- void [setOptions](#) (int options)
- int [options](#) () const
- virtual void [apply](#) ([QwtPlot](#) *) const
- virtual void [reset](#) ([QwtPlot](#) *) const
- virtual void [apply](#) ([QwtPlotItem](#) *) const
- virtual void [reset](#) ([QwtPlotItem](#) *) const

6.60.2 Member Enumeration Documentation

6.60.2.1 enum [QwtPlotPrintFilter::Options](#)

Print options.

Definition at line 34 of file qwt_plot_printfilter.h.

6.60.2.2 enum [QwtPlotPrintFilter::Item](#)

Print items.

Definition at line 47 of file qwt_plot_printfilter.h.

6.60.3 Constructor & Destructor Documentation

6.60.3.1 [QwtPlotPrintFilter::QwtPlotPrintFilter](#) () [explicit]

Sets filter options to PrintAll

Definition at line 82 of file qwt_plot_printfilter.cpp.

6.60.3.2 QwtPlotPrintFilter::~~QwtPlotPrintFilter () [virtual]

Destructor.

Definition at line 88 of file qwt_plot_printfilter.cpp.

6.60.4 Member Function Documentation

6.60.4.1 QColor QwtPlotPrintFilter::color (const QColor & *c*, [Item item](#)) const [virtual]

Modifies a color for printing.

Parameters:

c Color to be modified

item Type of item where the color belongs

Returns:

Modified color.

In case of `!(QwtPlotPrintFilter::options() & PrintBackground)` MajorGrid is modified to `Qt::darkGray`, MinorGrid to `Qt::gray`. All other colors are returned unmodified.

Definition at line 124 of file qwt_plot_printfilter.cpp.

References `options()`.

Referenced by `apply()`.

6.60.4.2 QFont QwtPlotPrintFilter::font (const QFont & *f*, [Item item](#)) const [virtual]

Modifies a font for printing.

Parameters:

f Font to be modified

item Type of item where the font belongs

All fonts are returned unmodified

Definition at line 148 of file qwt_plot_printfilter.cpp.

Referenced by `apply()`.

6.60.4.3 void QwtPlotPrintFilter::setOptions (int *options*)

Set plot print options.

Parameters:

options Or'd [QwtPlotPrintFilter::Options](#) values

See also:

[options\(\)](#)

Definition at line 99 of file qwt_plot_printfilter.cpp.

6.60.4.4 `int QwtPlotPrintFilter::options () const`

Get plot print options.

See also:

[setOptions\(\)](#)

Definition at line 108 of file `qwt_plot_printfilter.cpp`.

Referenced by `color()`, `QwtPlot::drawItems()`, `QwtPlot::print()`, and `QwtPlot::printCanvas()`.

6.60.4.5 `void QwtPlotPrintFilter::apply (QwtPlot *plot) const` [virtual]

Change color and fonts of a plot

See also:

[apply](#)

Definition at line 157 of file `qwt_plot_printfilter.cpp`.

References `QwtPlot::autoReplot()`, `QwtPlot::axisWidget()`, `QwtSymbol::brush()`, `QwtPlot::canvasBackground()`, `QwtText::color()`, `color()`, `QwtText::font()`, `font()`, `QwtPlotDict::itemList()`, `QwtPlot::legend()`, `QwtLegend::legendItems()`, `QwtSymbol::pen()`, `QwtPlot::setAutoReplot()`, `QwtSymbol::setBrush()`, `QwtPlot::setCanvasBackground()`, `QwtText::setColor()`, `QwtText::setFont()`, `QwtSymbol::setPen()`, `QwtText::testPaintAttribute()`, `QwtScaleWidget::title()`, and `QwtPlot::titleLabel()`.

Referenced by `QwtPlot::print()`.

6.60.4.6 `void QwtPlotPrintFilter::reset (QwtPlot *plot) const` [virtual]

Reset color and fonts of a plot

See also:

[apply](#)

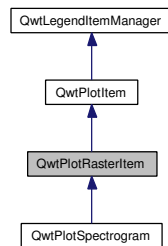
Definition at line 366 of file `qwt_plot_printfilter.cpp`.

References `QwtPlot::autoReplot()`, `QwtPlot::axisWidget()`, `QwtSymbol::brush()`, `QwtLegend::find()`, `QwtPlotDict::itemList()`, `QwtPlot::legend()`, `QwtLegend::legendItems()`, `QwtSymbol::pen()`, `QwtPlot::setAutoReplot()`, `QwtSymbol::setBrush()`, `QwtPlot::setCanvasBackground()`, `QwtText::setColor()`, `QwtSymbol::setPen()`, `QwtScaleWidget::setTitle()`, `QwtTextLabel::text()`, and `QwtPlot::titleLabel()`.

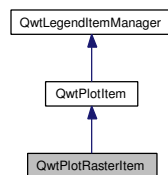
Referenced by `QwtPlot::print()`.

6.61 QwtPlotRasterItem Class Reference

Inheritance diagram for `QwtPlotRasterItem`:



Collaboration diagram for QwtPlotRasterItem:



6.61.1 Detailed Description

A class, which displays raster data.

Raster data is a grid of pixel values, that can be represented as a QImage. It is used for many types of information like spectrograms, cartograms, geographical maps ...

Often a plot has several types of raster data organized in layers. (f.e a geographical map, with weather statistics). Using [setAlpha\(\)](#) raster items can be stacked easily.

[QwtPlotRasterItem](#) is only implemented for images of the following formats: QImage::Format_Indexed8, QImage::Format_ARGB32.

See also:

[QwtPlotSpectrogram](#)

Definition at line 36 of file qwt_plot_rasteritem.h.

Public Types

- enum [CachePolicy](#) {
NoCache,
PaintCache,
ScreenCache }

Public Member Functions

- [QwtPlotRasterItem](#) (const QString &title=QString::null)
- [QwtPlotRasterItem](#) (const [QwtText](#) &title)
- virtual [~QwtPlotRasterItem](#) ()
- void [setAlpha](#) (int alpha)

- int `alpha` () const
- void `setCachePolicy` (CachePolicy)
- CachePolicy `cachePolicy` () const
- void `invalidateCache` ()
- virtual void `draw` (QPainter *p, const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QRect &rect) const
- virtual QSize `rasterHint` (const QwtDoubleRect &) const

Protected Member Functions

- virtual QImage `renderImage` (const QwtScaleMap &xMap, const QwtScaleMap &yMap, const QwtDoubleRect &area) const=0

6.61.2 Member Enumeration Documentation

6.61.2.1 enum QwtPlotRasterItem::CachePolicy

- NoCache
`renderImage()` is called, whenever the item has to be repainted
- PaintCache
`renderImage()` is called, whenever the image cache is not valid, or the scales, or the size of the canvas has changed. This type of cache is only useful for improving the performance of hide/show operations. All other situations are already handled by the plot canvas cache.
- ScreenCache
The screen cache is an image in size of the screen. As long as the scales don't change the target image is scaled from the cache. This might improve the performance when resizing the plot widget, but suffers from scaling effects.

The default policy is NoCache

Definition at line 56 of file `qwt_plot_rasteritem.h`.

6.61.3 Constructor & Destructor Documentation

6.61.3.1 QwtPlotRasterItem::QwtPlotRasterItem (const QString & title = QString::null) [explicit]

Constructor.

Definition at line 91 of file `qwt_plot_rasteritem.cpp`.

6.61.3.2 QwtPlotRasterItem::QwtPlotRasterItem (const QwtText & title) [explicit]

Constructor.

Definition at line 98 of file `qwt_plot_rasteritem.cpp`.

6.61.3.3 QwtPlotRasterItem::~QwtPlotRasterItem () [virtual]

Destructor.

Definition at line 105 of file `qwt_plot_rasteritem.cpp`.

6.61.4 Member Function Documentation

6.61.4.1 void QwtPlotRasterItem::setAlpha (int *alpha*)

Set an alpha value for the raster data.

Often a plot has several types of raster data organized in layers. (f.e a geographical map, with weather statistics). Using [setAlpha\(\)](#) raster items can be stacked easily.

The alpha value is a value [0, 255] to control the transparency of the image. 0 represents a fully transparent color, while 255 represents a fully opaque color.

Parameters:

alpha Alpha value

- $\alpha \geq 0$

All alpha values of the pixels returned by [renderImage\(\)](#) will be set to alpha, beside those with an alpha value of 0 (invalid pixels).

- $\alpha < 0$ The alpha values returned by [renderImage\(\)](#) are not changed.

The default alpha value is -1.

See also:

[alpha\(\)](#)

Definition at line 143 of file qwt_plot_rasteritem.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.61.4.2 int QwtPlotRasterItem::alpha () const

Returns:

Alpha value of the raster item

See also:

[setAlpha\(\)](#)

Definition at line 163 of file qwt_plot_rasteritem.cpp.

6.61.4.3 void QwtPlotRasterItem::setCachePolicy (QwtPlotRasterItem::CachePolicy *policy*)

Change the cache policy

The default policy is NoCache

Parameters:

policy Cache policy

See also:

[CachePolicy](#), [cachePolicy\(\)](#)

Definition at line 176 of file qwt_plot_rasteritem.cpp.

References [invalidateCache\(\)](#), and [QwtPlotItem::itemChanged\(\)](#).

6.61.4.4 QwtPlotRasterItem::CachePolicy QwtPlotRasterItem::cachePolicy () const**Returns:**

Cache policy

See also:

[CachePolicy](#), [setCachePolicy\(\)](#)

Definition at line 192 of file `qwt_plot_rasteritem.cpp`.

6.61.4.5 void QwtPlotRasterItem::invalidateCache ()

Invalidate the paint cache

See also:

[setCachePolicy](#)

Definition at line 201 of file `qwt_plot_rasteritem.cpp`.

Referenced by [setCachePolicy\(\)](#), [QwtPlotSpectrogram::setColorMap\(\)](#), and [QwtPlotSpectrogram::setData\(\)](#).

6.61.4.6 void QwtPlotRasterItem::draw (QPainter * *painter*, const [QwtScaleMap](#) & *xMap*, const [QwtScaleMap](#) & *yMap*, const QRect & *canvasRect*) const [virtual]

Draw the raster data.

Parameters:

painter Painter
xMap X-Scale Map
yMap Y-Scale Map
canvasRect Contents rect of the plot canvas

Implements [QwtPlotItem](#).

Reimplemented in [QwtPlotSpectrogram](#).

Definition at line 229 of file `qwt_plot_rasteritem.cpp`.

References [QwtPlotItem::boundingRect\(\)](#), [QwtPlotItem::invTransform\(\)](#), [QwtPlotItem::paintRect\(\)](#), [renderImage\(\)](#), [QwtScaleMap::setPaintInterval\(\)](#), and [QwtPlotItem::transform\(\)](#).

Referenced by [QwtPlotSpectrogram::draw\(\)](#).

6.61.4.7 QSize QwtPlotRasterItem::rasterHint (const [QwtDoubleRect](#) &) const [virtual]

Returns the recommended raster for a given rect.

E.g. the raster hint can be used to limit the resolution of the image that is rendered.

The default implementation returns an invalid size (`QSize()`), what means: no hint.

Reimplemented in [QwtPlotSpectrogram](#).

Definition at line 217 of file `qwt_plot_rasteritem.cpp`.

6.61.4.8 virtual QImage QwtPlotRasterItem::renderImage (const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, const QwtDoubleRect & *area*) const [protected, pure virtual]

Renders an image for an area

The format of the image must be QImage::Format_Indexed8, QImage::Format_RGB32 or QImage::Format_ARGB32

Parameters:

xMap Maps x-values into pixel coordinates.

yMap Maps y-values into pixel coordinates.

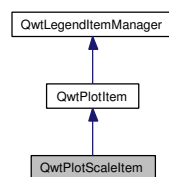
area Requested area for the image in scale coordinates

Implemented in [QwtPlotSpectrogram](#).

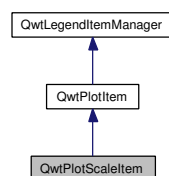
Referenced by draw().

6.62 QwtPlotScaleItem Class Reference

Inheritance diagram for QwtPlotScaleItem:



Collaboration diagram for QwtPlotScaleItem:



6.62.1 Detailed Description

A class which draws a scale inside the plot canvas.

[QwtPlotScaleItem](#) can be used to draw an axis inside the plot canvas. It might be synchronized to one of the axis of the plot, but can also display its own ticks and labels.

It is allowed to synchronize the scale item with a disabled axis. In plots with vertical and horizontal scale items, it might be necessary to remove ticks at the intersections, by overloading [updateScaleDiv\(\)](#).

The scale might be at a specific position (f.e 0.0) or it might be aligned to a canvas border.

Example

The following example shows how to replace the left axis, by a scale item at the x position 0.0.

```

QwtPlotScaleItem *scaleItem =
    new QwtPlotScaleItem(QwtScaleDraw::RightScale, 0.0);
scaleItem->setFont(plot->axisWidget(QwtPlot::yLeft)->font());
scaleItem->attach(plot);

plot->enableAxis(QwtPlot::yLeft, false);

```

Definition at line 51 of file qwt_plot_scaleitem.h.

Public Member Functions

- [QwtPlotScaleItem](#) ([QwtScaleDraw::Alignment](#)=[QwtScaleDraw::BottomScale](#), const double pos=0.0)
- virtual [~QwtPlotScaleItem](#) ()
- virtual int [rtti](#) () const
- void [setScaleDiv](#) (const [QwtScaleDiv](#) &)
- const [QwtScaleDiv](#) & [scaleDiv](#) () const
- void [setScaleDivFromAxis](#) (bool on)
- bool [isScaleDivFromAxis](#) () const
- void [setPalette](#) (const [QPalette](#) &)
- [QPalette](#) [palette](#) () const
- void [setFont](#) (const [QFont](#) &)
- [QFont](#) [font](#) () const
- void [setScaleDraw](#) ([QwtScaleDraw](#) *)
- const [QwtScaleDraw](#) * [scaleDraw](#) () const
- [QwtScaleDraw](#) * [scaleDraw](#) ()
- void [setPosition](#) (double pos)
- double [position](#) () const
- void [setBorderDistance](#) (int numPixels)
- int [borderDistance](#) () const
- void [setAlignment](#) ([QwtScaleDraw::Alignment](#))
- virtual void [draw](#) ([QPainter](#) *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QRect](#) &rect) const
- virtual void [updateScaleDiv](#) (const [QwtScaleDiv](#) &, const [QwtScaleDiv](#) &)

6.62.2 Constructor & Destructor Documentation

6.62.2.1 [QwtPlotScaleItem::QwtPlotScaleItem](#) ([QwtScaleDraw::Alignment](#) *alignment* = [QwtScaleDraw::BottomScale](#), const double *pos* = 0.0) [explicit]

Constructor for scale item at the position pos.

Parameters:

- alignment*** In case of [QwtScaleDraw::BottomScale](#)/[QwtScaleDrawTopScale](#) the scale item is corresponding to the [xAxis\(\)](#), otherwise it corresponds to the [yAxis\(\)](#).
- position*** x or y position, depending on the corresponding axis.

See also:

[setPosition\(\)](#), [setAlignment\(\)](#)

Definition at line 58 of file qwt_plot_scaleitem.cpp.

References [QwtPlotItem::setZ\(\)](#).

6.62.2.2 QwtPlotScaleItem::~~QwtPlotScaleItem () [virtual]

Destructor.

Definition at line 70 of file qwt_plot_scaleitem.cpp.

6.62.3 Member Function Documentation

6.62.3.1 int QwtPlotScaleItem::rtti () const [virtual]

Returns:

QwtPlotItem::Rtti_PlotScale

Reimplemented from [QwtPlotItem](#).

Definition at line 76 of file qwt_plot_scaleitem.cpp.

6.62.3.2 void QwtPlotScaleItem::setScaleDiv (const [QwtScaleDiv](#) & *scaleDiv*)

Assign a scale division.

When assigning a scaleDiv the scale division won't be synchronized with the corresponding axis anymore.

Parameters:

scaleDiv Scale division

See also:

[scaleDiv\(\)](#), [setScaleDivFromAxis\(\)](#), [isScaleDivFromAxis\(\)](#)

Definition at line 90 of file qwt_plot_scaleitem.cpp.

References [scaleDiv\(\)](#).

6.62.3.3 const [QwtScaleDiv](#) & QwtPlotScaleItem::scaleDiv () const

Returns:

Scale division

Definition at line 97 of file qwt_plot_scaleitem.cpp.

Referenced by [setScaleDiv\(\)](#).

6.62.3.4 void QwtPlotScaleItem::setScaleDivFromAxis (bool *on*)

Enable/Disable the synchronization of the scale division with the corresponding axis.

See also:

[isScaleDivFromAxis\(\)](#)

Definition at line 108 of file qwt_plot_scaleitem.cpp.

References [QwtPlot::axisScaleDiv\(\)](#), [QwtPlotItem::itemChanged\(\)](#), [QwtPlotItem::plot\(\)](#), [updateScaleDiv\(\)](#), [QwtPlotItem::xAxis\(\)](#), and [QwtPlotItem::yAxis\(\)](#).

6.62.3.5 bool QwtPlotScaleItem::isScaleDivFromAxis () const**Returns:**

True, if the synchronization of the scale division with the corresponding axis is enabled.

See also:

[setScaleDiv\(\)](#), [setScaleDivFromAxis\(\)](#)

Definition at line 131 of file qwt_plot_scaleitem.cpp.

6.62.3.6 void QwtPlotScaleItem::setPalette (const QPalette & *palette*)

Set the palette

See also:

[QwtAbstractScaleDraw::draw\(\)](#), [palette\(\)](#)

Definition at line 166 of file qwt_plot_scaleitem.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.62.3.7 QPalette QwtPlotScaleItem::palette () const**Returns:**

palette

See also:

[setPalette\(\)](#)

Definition at line 179 of file qwt_plot_scaleitem.cpp.

6.62.3.8 void QwtPlotScaleItem::setFont (const QFont & *font*)

Change the tick label font

See also:

[font](#)

Definition at line 190 of file qwt_plot_scaleitem.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.62.3.9 QFont QwtPlotScaleItem::font () const**Returns:**

tick label font

See also:

[setFont\(\)](#)

Definition at line 203 of file qwt_plot_scaleitem.cpp.

6.62.3.10 void QwtPlotScaleItem::setScaleDraw (QwtScaleDraw * scaleDraw)

Set a scale draw.

Parameters:

axisId axis index

scaleDraw object responsible for drawing scales.

The main use case for replacing the default [QwtScaleDraw](#) is to overload [QwtAbstractScaleDraw::label](#), to replace or swallow tick labels.

See also:

[scaleDraw\(\)](#)

Definition at line 220 of file `qwt_plot_scaleitem.cpp`.

References [QwtPlot::axisScaleDiv\(\)](#), [QwtPlotItem::itemChanged\(\)](#), [QwtPlotItem::plot\(\)](#), [scaleDraw\(\)](#), [updateScaleDiv\(\)](#), [QwtPlotItem::xAxis\(\)](#), and [QwtPlotItem::yAxis\(\)](#).

6.62.3.11 const QwtScaleDraw * QwtPlotScaleItem::scaleDraw () const**Returns:**

Scale draw

See also:

[setScaleDraw\(\)](#)

Definition at line 244 of file `qwt_plot_scaleitem.cpp`.

Referenced by [setScaleDraw\(\)](#).

6.62.3.12 QwtScaleDraw * QwtPlotScaleItem::scaleDraw ()**Returns:**

Scale draw

See also:

[setScaleDraw\(\)](#)

Definition at line 253 of file `qwt_plot_scaleitem.cpp`.

6.62.3.13 void QwtPlotScaleItem::setPosition (double pos)

Change the position of the scale

The position is interpreted as y value for horizontal axes and as x value for vertical axes.

The border distance is set to -1.

See also:

[position\(\)](#), [setAlignment\(\)](#)

Definition at line 268 of file `qwt_plot_scaleitem.cpp`.

References [QwtPlotItem::itemChanged\(\)](#).

6.62.3.14 double QwtPlotScaleItem::position () const**Returns:**

Position of the scale

See also:

[setPosition\(\)](#), [setAlignment\(\)](#)

Definition at line 282 of file qwt_plot_scaleitem.cpp.

6.62.3.15 void QwtPlotScaleItem::setBorderDistance (int *distance*)

Align the scale to the canvas.

If distance is ≥ 0 the scale will be aligned to a border of the contents rect of the canvas. If alignment() is QwtScaleDraw::LeftScale, the scale will be aligned to the right border, if it is QwtScaleDraw::TopScale it will be aligned to the bottom (and vice versa),

If distance is < 0 the scale will be at the [position\(\)](#).

Parameters:

distance Number of pixels between the canvas border and the backbone of the scale.

See also:

[setPosition\(\)](#), [borderDistance\(\)](#)

Definition at line 303 of file qwt_plot_scaleitem.cpp.

References QwtPlotItem::itemChanged().

6.62.3.16 int QwtPlotScaleItem::borderDistance () const**Returns:**

Distance from a canvas border

See also:

[setBorderDistance\(\)](#), [setPosition\(\)](#)

Definition at line 319 of file qwt_plot_scaleitem.cpp.

6.62.3.17 void QwtPlotScaleItem::setAlignment (QwtScaleDraw::Alignment *alignment*)

Change the alignment of the scale

The alignment sets the orientation of the scale and the position of the ticks:

- QwtScaleDraw::BottomScale: horizontal, ticks below
- QwtScaleDraw::TopScale: horizontal, ticks above
- QwtScaleDraw::LeftScale: vertical, ticks left

- `QwtScaleDraw::RightScale`: vertical, ticks right

For horizontal scales the position corresponds to `QwtPlotItem::yAxis()`, otherwise to `QwtPlotItem::xAxis()`.

See also:

`scaleDraw()`, `QwtScaleDraw::alignment()`, `setPosition()`

Definition at line 340 of file `qwt_plot_scaleitem.cpp`.

References `QwtScaleDraw::alignment()`, `QwtPlotItem::itemChanged()`, and `QwtScaleDraw::setAlignment()`.

6.62.3.18 `void QwtPlotScaleItem::draw (QPainter * p, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRect & rect) const` [virtual]

Draw the scale.

Implements [QwtPlotItem](#).

Definition at line 353 of file `qwt_plot_scaleitem.cpp`.

References `QwtScaleDraw::alignment()`, `QwtScaleTransformation::copy()`, `QwtAbstractScaleDraw::draw()`, `QwtScaleDraw::move()`, `QwtScaleDraw::orientation()`, `QwtScaleDraw::setLength()`, `QwtAbstractScaleDraw::setTransformation()`, `QwtScaleMap::transform()`, `QwtScaleMap::transformation()`, and `updateBorders()`.

6.62.3.19 `void QwtPlotScaleItem::updateScaleDiv (const QwtScaleDiv & xScaleDiv, const QwtScaleDiv & yScaleDiv)` [virtual]

Update the item to changes of the axes scale division.

In case of `isScaleDivFromAxis()`, the scale draw is synchronized to the correspond axis.

Parameters:

xScaleDiv Scale division of the x-axis

yScaleDiv Scale division of the y-axis

See also:

[QwtPlot::updateAxes\(\)](#)

Reimplemented from [QwtPlotItem](#).

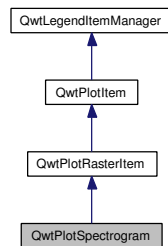
Definition at line 443 of file `qwt_plot_scaleitem.cpp`.

References `QwtScaleDraw::orientation()`, and `QwtAbstractScaleDraw::setScaleDiv()`.

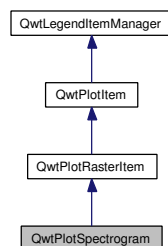
Referenced by `setScaleDivFromAxis()`, and `setScaleDraw()`.

6.63 QwtPlotSpectrogram Class Reference

Inheritance diagram for `QwtPlotSpectrogram`:



Collaboration diagram for QwtPlotSpectrogram:



6.63.1 Detailed Description

A plot item, which displays a spectrogram.

A spectrogram displays threedimensional data, where the 3rd dimension (the intensity) is displayed using colors. The colors are calculated from the values using a color map.

In ContourMode contour lines are painted for the contour levels.

See also:

[QwtRasterData](#), [QwtColorMap](#)

Definition at line 35 of file `qwt_plot_spectrogram.h`.

Public Types

- enum [DisplayMode](#) {
 AlwaysOff,
 AlwaysOn,
 ActiveOnly,
 ImageMode = 1,
 ContourMode = 2 }

Public Member Functions

- [QwtPlotSpectrogram](#) (const QString &title=QString::null)
- virtual [~QwtPlotSpectrogram](#) ()
- void [setDisplayMode](#) ([DisplayMode](#), bool on=true)

- bool [testDisplayMode](#) ([DisplayMode](#)) const
- void [setData](#) (const [QwtRasterData](#) &data)
- const [QwtRasterData](#) & [data](#) () const
- void [setColorMap](#) (const [QwtColorMap](#) &)
- const [QwtColorMap](#) & [colorMap](#) () const
- virtual [QwtDoubleRect](#) [boundingRect](#) () const
- virtual QSize [rasterHint](#) (const [QwtDoubleRect](#) &) const
- void [setDefaultContourPen](#) (const [QPen](#) &)
- [QPen](#) [defaultContourPen](#) () const
- virtual [QPen](#) [contourPen](#) (double level) const
- void [setConrecAttribute](#) ([QwtRasterData::ConrecAttribute](#), bool on)
- bool [testConrecAttribute](#) ([QwtRasterData::ConrecAttribute](#)) const
- void [setContourLevels](#) (const [QwtValueList](#) &)
- [QwtValueList](#) [contourLevels](#) () const
- virtual int [rtti](#) () const
- virtual void [draw](#) ([QPainter](#) *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QRect](#) &rect) const

Protected Member Functions

- virtual [QImage](#) [renderImage](#) (const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QwtDoubleRect](#) &rect) const
- virtual QSize [contourRasterSize](#) (const [QwtDoubleRect](#) &, const [QRect](#) &) const
- virtual [QwtRasterData::ContourLines](#) [renderContourLines](#) (const [QwtDoubleRect](#) &rect, const QSize &raster) const
- virtual void [drawContourLines](#) ([QPainter](#) *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const [QwtRasterData::ContourLines](#) &lines) const

6.63.2 Member Enumeration Documentation

6.63.2.1 enum [QwtPlotSpectrogram::DisplayMode](#)

The display mode controls how the raster data will be represented.

- [ImageMode](#)
The values are mapped to colors using a color map.
- [ContourMode](#)
The data is displayed using contour lines

When both modes are enabled the contour lines are painted on top of the spectrogram. The default setting enables [ImageMode](#).

See also:

[setDisplayMode\(\)](#), [testDisplayMode\(\)](#)

Definition at line 51 of file `qwt_plot_spectrogram.h`.

6.63.3 Constructor & Destructor Documentation

6.63.3.1 QwtPlotSpectrogram::QwtPlotSpectrogram (const QString & *title* = QString::null) [explicit]

Sets the following item attributes:

- QwtPlotItem::AutoScale: true
- QwtPlotItem::Legend: false

The z value is initialized by 8.0.

Parameters:

title Title

See also:

[QwtPlotItem::setItemAttribute\(\)](#), [QwtPlotItem::setZ\(\)](#)

Definition at line 134 of file qwt_plot_spectrogram.cpp.

References [QwtPlotItem::setItemAttribute\(\)](#), and [QwtPlotItem::setZ\(\)](#).

6.63.3.2 QwtPlotSpectrogram::~~QwtPlotSpectrogram () [virtual]

Destructor.

Definition at line 146 of file qwt_plot_spectrogram.cpp.

6.63.4 Member Function Documentation

6.63.4.1 void QwtPlotSpectrogram::setDisplayMode (DisplayMode *mode*, bool *on* = true)

The display mode controls how the raster data will be represented.

Parameters:

mode Display mode

on On/Off

The default setting enables ImageMode.

See also:

[DisplayMode](#), [displayMode\(\)](#)

Definition at line 167 of file qwt_plot_spectrogram.cpp.

References [QwtPlotItem::itemChanged\(\)](#).

6.63.4.2 bool QwtPlotSpectrogram::testDisplayMode (DisplayMode *mode*) const

The display mode controls how the raster data will be represented.

Parameters:

mode Display mode

Returns:

true if mode is enabled

Definition at line 186 of file qwt_plot_spectrogram.cpp.

6.63.4.3 void QwtPlotSpectrogram::setData (const [QwtRasterData](#) & data)

Set the data to be displayed

Parameters:

data Spectrogram Data

See also:

[data\(\)](#)

Definition at line 342 of file qwt_plot_spectrogram.cpp.

References [QwtRasterData::copy\(\)](#), [data\(\)](#), [QwtPlotRasterItem::invalidateCache\(\)](#), and [QwtPlotItem::itemChanged\(\)](#).

6.63.4.4 const [QwtRasterData](#) & QwtPlotSpectrogram::data () const**Returns:**

Spectrogram data

See also:

[setData\(\)](#)

Definition at line 355 of file qwt_plot_spectrogram.cpp.

Referenced by [setData\(\)](#).

6.63.4.5 void QwtPlotSpectrogram::setColorMap (const [QwtColorMap](#) & colorMap)

Change the color map

Often it is useful to display the mapping between intensities and colors as an additional plot axis, showing a color bar.

Parameters:

colorMap Color Map

See also:

[colorMap\(\)](#), [QwtScaleWidget::setColorBarEnabled\(\)](#), [QwtScaleWidget::setColorMap\(\)](#)

Definition at line 202 of file qwt_plot_spectrogram.cpp.

References [colorMap\(\)](#), [QwtColorMap::copy\(\)](#), [QwtPlotRasterItem::invalidateCache\(\)](#), and [QwtPlotItem::itemChanged\(\)](#).

6.63.4.6 `const QwtColorMap & QwtPlotSpectrogram::colorMap () const`

Returns:

Color Map used for mapping the intensity values to colors

See also:

[setColorMap\(\)](#)

Definition at line 215 of file `qwt_plot_spectrogram.cpp`.

Referenced by `setColorMap()`.

6.63.4.7 `QwtDoubleRect QwtPlotSpectrogram::boundingRect () const` [virtual]

Returns:

Bounding rect of the data

See also:

[QwtRasterData::boundingRect](#)

Reimplemented from [QwtPlotItem](#).

Definition at line 364 of file `qwt_plot_spectrogram.cpp`.

Referenced by `draw()`.

6.63.4.8 `QSize QwtPlotSpectrogram::rasterHint (const QwtDoubleRect & rect) const` [virtual]

Returns the recommended raster for a given rect.

E.g the raster hint is used to limit the resolution of the image that is rendered.

Parameters:

rect Rect for the raster hint

Returns:

`data().rasterHint(rect)`

Reimplemented from [QwtPlotRasterItem](#).

Definition at line 378 of file `qwt_plot_spectrogram.cpp`.

Referenced by `contourRasterSize()`.

6.63.4.9 `void QwtPlotSpectrogram::setDefaultContourPen (const QPen & pen)`

Set the default pen for the contour lines.

If the spectrogram has a valid default contour pen a contour line is painted using the default contour pen. Otherwise (`pen.style() == Qt::NoPen`) the pen is calculated for each contour level using [contourPen\(\)](#).

See also:

[defaultContourPen](#), [contourPen](#)

Definition at line 230 of file qwt_plot_spectrogram.cpp.

References `QwtPlotItem::itemChanged()`.

6.63.4.10 QPen QwtPlotSpectrogram::defaultContourPen () const

Returns:

Default contour pen

See also:

[setDefaultContourPen](#)

Definition at line 243 of file qwt_plot_spectrogram.cpp.

Referenced by `drawContourLines()`.

6.63.4.11 QPen QwtPlotSpectrogram::contourPen (double *level*) const [virtual]

Calculate the pen for a contour line.

The color of the pen is the color for level calculated by the color map

Parameters:

level Contour level

Returns:

Pen for the contour line

Note:

`contourPen` is only used if `defaultContourPen().style() == Qt::NoPen`

See also:

[setDefaultContourPen](#), [setColorMap](#), [setContourLevels](#)

Definition at line 259 of file qwt_plot_spectrogram.cpp.

Referenced by `drawContourLines()`.

6.63.4.12 void QwtPlotSpectrogram::setConrecAttribute (QwtRasterData::ConrecAttribute *attribute*, bool *on*)

Modify an attribute of the CONREC algorithm, used to calculate the contour lines.

Parameters:

attribute CONREC attribute

on On/Off

See also:

[testConrecAttribute](#), [renderContourLines](#), [QwtRasterData::contourLines](#)

Definition at line 276 of file `qwt_plot_spectrogram.cpp`.

References `QwtPlotItem::itemChanged()`.

6.63.4.13 `bool QwtPlotSpectrogram::testConrecAttribute (QwtRasterData::ConrecAttribute attribute) const`

Test an attribute of the CONREC algorithm, used to calculate the contour lines.

Parameters:

attribute CONREC attribute

Returns:

true, is enabled

See also:

[setConrecAttribute](#), [renderContourLines](#), [QwtRasterData::contourLines](#)

Definition at line 299 of file `qwt_plot_spectrogram.cpp`.

6.63.4.14 `void QwtPlotSpectrogram::setContourLevels (const QwtValueList & levels)`

Set the levels of the contour lines

Parameters:

levels Values of the contour levels

See also:

[contourLevels](#), [renderContourLines](#), [QwtRasterData::contourLines](#)

Note:

`contourLevels` returns the same levels but sorted.

Definition at line 313 of file `qwt_plot_spectrogram.cpp`.

References `QwtPlotItem::itemChanged()`.

6.63.4.15 `QwtValueList QwtPlotSpectrogram::contourLevels () const`

Return the levels of the contour lines.

The levels are sorted in increasing order.

See also:

[contourLevels](#), [renderContourLines](#), [QwtRasterData::contourLines](#)

Definition at line 331 of file `qwt_plot_spectrogram.cpp`.

6.63.4.16 `int QwtPlotSpectrogram::rtti () const` [virtual]**Returns:**

`QwtPlotItem::Rtti_PlotSpectrogram`

Reimplemented from [QwtPlotItem](#).

Definition at line 152 of file `qwt_plot_spectrogram.cpp`.

6.63.4.17 `void QwtPlotSpectrogram::draw (QPainter *painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRect & canvasRect) const` [virtual]

Draw the spectrogram.

Parameters:

painter Painter

xMap Maps x-values into pixel coordinates.

yMap Maps y-values into pixel coordinates.

canvasRect Contents rect of the canvas in painter coordinates

See also:

[setDisplayMode](#), [renderImage](#), [QwtPlotRasterItem::draw](#), [drawContourLines](#)

Reimplemented from [QwtPlotRasterItem](#).

Definition at line 615 of file `qwt_plot_spectrogram.cpp`.

References [boundingRect\(\)](#), [contourRasterSize\(\)](#), [QwtPlotRasterItem::draw\(\)](#), [drawContourLines\(\)](#), [QwtPlotItem::invTransform\(\)](#), [renderContourLines\(\)](#), and [QwtPlotItem::transform\(\)](#).

6.63.4.18 `QImage QwtPlotSpectrogram::renderImage (const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QwtDoubleRect & area) const` [protected, virtual]

Render an image from the data and color map.

The area is translated into a rect of the paint device. For each pixel of this rect the intensity is mapped into a color.

Parameters:

xMap X-Scale Map

yMap Y-Scale Map

area Area that should be rendered in scale coordinates.

Returns:

A `QImage::Format_Indexed8` or `QImage::Format_ARGB32` depending on the color map.

See also:

[QwtRasterData::intensity\(\)](#), [QwtColorMap::rgb\(\)](#), [QwtColorMap::colorIndex\(\)](#)

Implements [QwtPlotRasterItem](#).

Definition at line 400 of file qwt_plot_spectrogram.cpp.

References [QwtScaleMap::invTransform\(\)](#), [QwtScaleMap::p1\(\)](#), [QwtScaleMap::p2\(\)](#), [QwtScaleMap::s1\(\)](#), [QwtScaleMap::s2\(\)](#), [QwtScaleMap::setPaintInterval\(\)](#), [QwtScaleMap::setScaleInterval\(\)](#), and [QwtPlotItem::transform\(\)](#).

6.63.4.19 QSize QwtPlotSpectrogram::contourRasterSize (const [QwtDoubleRect](#) & *area*, const [QRect](#) & *rect*) const [protected, virtual]

Return the raster to be used by the CONREC contour algorithm.

A larger size will improve the precision of the CONREC algorithm, but will slow down the time that is needed to calculate the lines.

The default implementation returns `rect.size() / 2` bounded to [data\(\).rasterHint\(\)](#).

Parameters:

- area* Rect, where to calculate the contour lines
- rect* Rect in pixel coordinates, where to paint the contour lines

Returns:

Raster to be used by the CONREC contour algorithm.

Note:

The size will be bounded to `rect.size()`.

See also:

[drawContourLines](#), [QwtRasterData::contourLines](#)

Definition at line 529 of file qwt_plot_spectrogram.cpp.

References [rasterHint\(\)](#).

Referenced by [draw\(\)](#).

6.63.4.20 [QwtRasterData::ContourLines](#) QwtPlotSpectrogram::renderContourLines (const [QwtDoubleRect](#) & *rect*, const [QSize](#) & *raster*) const [protected, virtual]

Calculate contour lines

Parameters:

- rect* Rectangle, where to calculate the contour lines
- raster* Raster, used by the CONREC algorithm

See also:

[contourLevels](#), [setConrecAttribute](#), [QwtRasterData::contourLines](#)

Definition at line 549 of file qwt_plot_spectrogram.cpp.

Referenced by [draw\(\)](#).

6.63.4.21 void QwtPlotSpectrogram::drawContourLines (QPainter * *painter*, const QwtScaleMap & *xMap*, const QwtScaleMap & *yMap*, const QwtRasterData::ContourLines & *contourLines*) const [protected, virtual]

Paint the contour lines

Parameters:

painter Painter

xMap Maps x-values into pixel coordinates.

yMap Maps y-values into pixel coordinates.

contourLines Contour lines

See also:

[renderContourLines](#), [defaultContourPen](#), [contourPen](#)

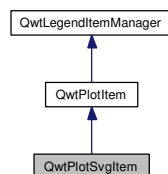
Definition at line 566 of file qwt_plot_spectrogram.cpp.

References [contourPen\(\)](#), [defaultContourPen\(\)](#), [QwtPainter::drawLine\(\)](#), and [QwtScaleMap::transform\(\)](#).

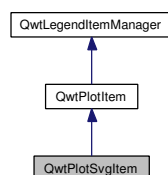
Referenced by [draw\(\)](#).

6.64 QwtPlotSvgItem Class Reference

Inheritance diagram for QwtPlotSvgItem:



Collaboration diagram for QwtPlotSvgItem:



6.64.1 Detailed Description

A plot item, which displays data in Scalable Vector Graphics (SVG) format.

SVG images are often used to display maps

Definition at line 31 of file qwt_plot_svgitem.h.

Public Member Functions

- [QwtPlotSvgItem](#) (const QString &title=QString::null)
- [QwtPlotSvgItem](#) (const [QwtText](#) &title)
- virtual [~QwtPlotSvgItem](#) ()
- bool [loadFile](#) (const [QwtDoubleRect](#) &, const QString &fileName)
- bool [loadData](#) (const [QwtDoubleRect](#) &, const QByteArray &)
- virtual [QwtDoubleRect boundingRect](#) () const
- virtual void [draw](#) (QPainter *p, const [QwtScaleMap](#) &xMap, const [QwtScaleMap](#) &yMap, const QRect &rect) const
- virtual int [rtti](#) () const

Protected Member Functions

- void [render](#) (QPainter *painter, const [QwtDoubleRect](#) &viewBox, const QRect &rect) const
- [QwtDoubleRect viewBox](#) (const [QwtDoubleRect](#) &area) const

6.64.2 Constructor & Destructor Documentation**6.64.2.1 QwtPlotSvgItem::QwtPlotSvgItem (const QString & title = QString::null) [explicit]**

Constructor.

Sets the following item attributes:

- QwtPlotItem::AutoScale: true
- QwtPlotItem::Legend: false

Parameters:

title Title

Definition at line 51 of file qwt_plot_svgitem.cpp.

6.64.2.2 QwtPlotSvgItem::QwtPlotSvgItem (const QwtText & title) [explicit]

Constructor.

Sets the following item attributes:

- QwtPlotItem::AutoScale: true
- QwtPlotItem::Legend: false

Parameters:

title Title

Definition at line 66 of file qwt_plot_svgitem.cpp.

6.64.2.3 QwtPlotSvgItem::~~QwtPlotSvgItem () [virtual]

Destructor.

Definition at line 73 of file qwt_plot_svgitem.cpp.

6.64.3 Member Function Documentation

6.64.3.1 bool QwtPlotSvgItem::loadFile (const QwtDoubleRect & rect, const QString & fileName)

Load a SVG file

Parameters:

rect Bounding rectangle
fileName SVG file name

Returns:

true, if the SVG file could be loaded

Definition at line 102 of file qwt_plot_svgitem.cpp.

References QwtPlotItem::itemChanged().

6.64.3.2 bool QwtPlotSvgItem::loadData (const QwtDoubleRect & rect, const QByteArray & data)

Load SVG data

Parameters:

rect Bounding rectangle
data in SVG format

Returns:

true, if the SVG data could be loaded

Definition at line 123 of file qwt_plot_svgitem.cpp.

References QwtPlotItem::itemChanged().

6.64.3.3 QwtDoubleRect QwtPlotSvgItem::boundingRect () const [virtual]

Bounding rect of the item.

Reimplemented from [QwtPlotItem](#).

Definition at line 142 of file qwt_plot_svgitem.cpp.

Referenced by draw(), and viewBox().

6.64.3.4 void QwtPlotSvgItem::draw (QPainter * painter, const QwtScaleMap & xMap, const QwtScaleMap & yMap, const QRect & canvasRect) const [virtual]

Draw the SVG item

Parameters:

painter Painter
xMap X-Scale Map
yMap Y-Scale Map
canvasRect Contents rect of the plot canvas

Implements [QwtPlotItem](#).

Definition at line 170 of file qwt_plot_svgitem.cpp.

References [boundingRect\(\)](#), [QwtPlotItem::invTransform\(\)](#), [render\(\)](#), [QwtPlotItem::transform\(\)](#), and [viewBox\(\)](#).

6.64.3.5 int QwtPlotSvgItem::rtti () const [virtual]**Returns:**

[QwtPlotItem::Rtti_PlotSVG](#)

Reimplemented from [QwtPlotItem](#).

Definition at line 89 of file qwt_plot_svgitem.cpp.

6.64.3.6 void QwtPlotSvgItem::render (QPainter *painter, const QwtDoubleRect &viewBox, const QRect &rect) const [protected]

Render the SVG data

Parameters:

painter Painter
viewBox View Box, see [QSvgRenderer::viewBox](#)
rect Target rectangle on the paint device

Definition at line 194 of file qwt_plot_svgitem.cpp.

Referenced by [draw\(\)](#).

6.64.3.7 QwtDoubleRect QwtPlotSvgItem::viewBox (const QwtDoubleRect &rect) const [protected]

Calculate the viewBox from an rect and [boundingRect\(\)](#).

Parameters:

rect Rectangle in scale coordinates

Returns:

[viewBox](#) View Box, see [QSvgRenderer::viewBox](#)

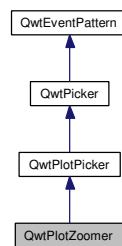
Definition at line 254 of file qwt_plot_svgitem.cpp.

References [boundingRect\(\)](#), [QwtScaleMap::setPaintInterval\(\)](#), [QwtScaleMap::setScaleInterval\(\)](#), and [QwtScaleMap::xTransform\(\)](#).

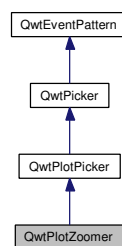
Referenced by [draw\(\)](#).

6.65 QwtPlotZoomer Class Reference

Inheritance diagram for QwtPlotZoomer:



Collaboration diagram for QwtPlotZoomer:



6.65.1 Detailed Description

[QwtPlotZoomer](#) provides stacked zooming for a plot widget.

[QwtPlotZoomer](#) offers rubberband selections on the plot canvas, translating the selected rectangles into plot coordinates and adjusting the axes to them. Zooming can be repeated as often as possible, limited only by [maxStackDepth\(\)](#) or [minZoomSize\(\)](#). Each rectangle is pushed on a stack.

Zoom rectangles can be selected depending on [selectionFlags\(\)](#) using the mouse or keyboard ([QwtEventPattern](#), [QwtPickerMachine](#)). [QwtEventPattern::MouseSelect3/QwtEventPatternKeyUndo](#), or [QwtEventPattern::MouseSelect6/QwtEventPatternKeyRedo](#) walk up and down the zoom stack. [QwtEventPattern::MouseSelect2](#) or [QwtEventPattern::KeyHome](#) unzoom to the initial size.

[QwtPlotZoomer](#) is tailored for plots with one x and y axis, but it is allowed to attach a second [QwtPlotZoomer](#) for the other axes.

Note:

The realtime example includes an derived zoomer class that adds scrollbars to the plot canvas.

Definition at line 49 of file `qwt_plot_zoomer.h`.

Public Slots

- void [moveBy](#) (double x, double y)
- virtual void [move](#) (double x, double y)
- virtual void [zoom](#) (const [QwtDoubleRect](#) &)
- virtual void [zoom](#) (int up)

Signals

- void [zoomed](#) (const [QwtDoubleRect](#) &rect)

Public Member Functions

- [QwtPlotZoomer](#) ([QwtPlotCanvas](#) *, bool doReplot=true)
- [QwtPlotZoomer](#) (int xAxis, int yAxis, [QwtPlotCanvas](#) *, bool doReplot=true)
- [QwtPlotZoomer](#) (int xAxis, int yAxis, int selectionFlags, [DisplayMode](#) trackerMode, [QwtPlotCanvas](#) *, bool doReplot=true)
- virtual [~QwtPlotZoomer](#) ()
- virtual void [setZoomBase](#) (bool doReplot=true)
- virtual void [setZoomBase](#) (const [QwtDoubleRect](#) &)
- [QwtDoubleRect](#) [zoomBase](#) () const
- [QwtDoubleRect](#) [zoomRect](#) () const
- virtual void [setAxis](#) (int xAxis, int yAxis)
- void [setMaxStackDepth](#) (int)
- int [maxStackDepth](#) () const
- const [QStack](#)< [QwtDoubleRect](#) > & [zoomStack](#) () const
- void [setZoomStack](#) (const [QStack](#)< [QwtDoubleRect](#) > &, int zoomRectIndex=-1)
- uint [zoomRectIndex](#) () const
- virtual void [setSelectionFlags](#) (int)

Protected Member Functions

- virtual void [rescale](#) ()
- virtual [QwtDoubleSize](#) [minZoomSize](#) () const
- virtual void [widgetMouseEvent](#) (QMouseEvent *)
- virtual void [widgetKeyPressEvent](#) (QKeyEvent *)
- virtual void [begin](#) ()
- virtual bool [end](#) (bool ok=true)
- virtual bool [accept](#) ([QwtPolygon](#) &) const

6.65.2 Constructor & Destructor Documentation

6.65.2.1 [QwtPlotZoomer::QwtPlotZoomer](#) ([QwtPlotCanvas](#) * *canvas*, bool *doReplot* = true) [explicit]

Create a zoomer for a plot canvas.

The zoomer is set to those x- and y-axis of the parent plot of the canvas that are enabled. If both or no x-axis are enabled, the picker is set to [QwtPlot::xBottom](#). If both or no y-axis are enabled, it is set to [QwtPlot::yLeft](#).

The [selectionFlags\(\)](#) are set to [QwtPicker::RectSelection](#) & [QwtPicker::ClickSelection](#), the tracker mode to [QwtPicker::ActiveOnly](#).

Parameters:

canvas Plot canvas to observe, also the parent object

doReplot Call replot for the attached plot before initializing the zoomer with its scales. This might be necessary, when the plot is in a state with pending scale changes.

See also:

[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [setZoomBase\(\)](#)

Definition at line 51 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::canvas\(\)](#).

6.65.2.2 QwtPlotZoomer::QwtPlotZoomer (int xAxis, int yAxis, [QwtPlotCanvas](#) * canvas, bool doReplot = true) [explicit]

Create a zoomer for a plot canvas.

The [selectionFlags\(\)](#) are set to [QwtPicker::RectSelection](#) & [QwtPicker::ClickSelection](#), the tracker mode to [QwtPicker::ActiveOnly](#).

Parameters:

xAxis X axis of the zoomer

yAxis Y axis of the zoomer

canvas Plot canvas to observe, also the parent object

doReplot Call replot for the attached plot before initializing the zoomer with its scales. This might be necessary, when the plot is in a state with pending scale changes.

See also:

[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [setZoomBase\(\)](#)

Definition at line 75 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::canvas\(\)](#).

6.65.2.3 QwtPlotZoomer::QwtPlotZoomer (int xAxis, int yAxis, int selectionFlags, [DisplayMode](#) trackerMode, [QwtPlotCanvas](#) * canvas, bool doReplot = true) [explicit]

Create a zoomer for a plot canvas.

Parameters:

xAxis X axis of the zoomer

yAxis Y axis of the zoomer

selectionFlags Or'd value of [QwtPicker::RectSelectionType](#) and [QwtPicker::SelectionMode](#). [QwtPicker::RectSelection](#) will be auto added.

trackerMode Tracker mode

canvas Plot canvas to observe, also the parent object

doReplot Call replot for the attached plot before initializing the zoomer with its scales. This might be necessary, when the plot is in a state with pending scale changes.

See also:

[QwtPicker](#), [QwtPicker::setSelectionFlags\(\)](#), [QwtPicker::setRubberBand\(\)](#), [QwtPicker::setTrackerMode](#)
[QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#), [setZoomBase\(\)](#)

Definition at line 103 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::canvas\(\)](#).

6.65.3 Member Function Documentation

6.65.3.1 void QwtPlotZoomer::setZoomBase (bool *doReplot* = true) [virtual]

Reinitialized the zoom stack with [scaleRect\(\)](#) as base.

Parameters:

doReplot Call replot for the attached plot before initializing the zoomer with its scales. This might be necessary, when the plot is in a state with pending scale changes.

See also:

[zoomBase\(\)](#), [scaleRect\(\)](#) [QwtPlot::autoReplot\(\)](#), [QwtPlot::replot\(\)](#).

Definition at line 206 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::plot\(\)](#), [QwtPlot::replot\(\)](#), [rescale\(\)](#), and [QwtPlotPicker::scaleRect\(\)](#).

Referenced by [setAxis\(\)](#).

6.65.3.2 void QwtPlotZoomer::setZoomBase (const [QwtDoubleRect](#) & *base*) [virtual]

Set the initial size of the zoomer.

base is united with the current [scaleRect\(\)](#) and the zoom stack is reinitialized with it as zoom base. plot is zoomed to [scaleRect\(\)](#).

Parameters:

base Zoom base

See also:

[zoomBase\(\)](#), [scaleRect\(\)](#)

Definition at line 232 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::plot\(\)](#), [rescale\(\)](#), and [QwtPlotPicker::scaleRect\(\)](#).

6.65.3.3 [QwtDoubleRect](#) QwtPlotZoomer::zoomBase () const

Returns:

Initial rectangle of the zoomer

See also:

[setZoomBase\(\)](#), [zoomRect\(\)](#)

Definition at line 192 of file `qwt_plot_zoomer.cpp`.

Referenced by [move\(\)](#).

6.65.3.4 [QwtDoubleRect](#) QwtPlotZoomer::zoomRect () const

Rectangle at the current position on the zoom stack.

See also:

[zoomRectIndex\(\)](#), [scaleRect\(\)](#).

Definition at line 259 of file `qwt_plot_zoomer.cpp`.

Referenced by `end()`, `move()`, and `zoom()`.

6.65.3.5 void QwtPlotZoomer::setAxis (int *xAxis*, int *yAxis*) [virtual]

Reinitialize the axes, and set the zoom base to their scales.

Parameters:

xAxis X axis

yAxis Y axis

Reimplemented from [QwtPlotPicker](#).

Definition at line 429 of file `qwt_plot_zoomer.cpp`.

References `QwtPlotPicker::scaleRect()`, `QwtPlotPicker::setAxis()`, `setZoomBase()`, `QwtPlotPicker::xAxis()`, and `QwtPlotPicker::yAxis()`.

6.65.3.6 void QwtPlotZoomer::setMaxStackDepth (int *depth*)

Limit the number of recursive zoom operations to depth.

A value of -1 set the depth to unlimited, 0 disables zooming. If the current zoom rectangle is below depth, the plot is unzoomed.

Parameters:

depth Maximum for the stack depth

See also:

[maxStackDepth\(\)](#)

Note:

depth doesn't include the zoom base, so `zoomStack().count()` might be `maxStackDepth() + 1`.

Definition at line 145 of file `qwt_plot_zoomer.cpp`.

References `zoom()`.

6.65.3.7 int QwtPlotZoomer::maxStackDepth () const

Returns:

Maximal depth of the zoom stack.

See also:

[setMaxStackDepth\(\)](#)

Definition at line 172 of file `qwt_plot_zoomer.cpp`.

6.65.3.8 const QwtZoomStack & QwtPlotZoomer::zoomStack () const

Return the zoom stack. [zoomStack\(\)\[0\]](#) is the zoom base, [zoomStack\(\)\[1\]](#) the first zoomed rectangle.

See also:

[setZoomStack\(\)](#), [zoomRectIndex\(\)](#)

Definition at line 183 of file `qwt_plot_zoomer.cpp`.

6.65.3.9 uint QwtPlotZoomer::zoomRectIndex () const

Returns:

Index of current position of zoom stack.

Definition at line 267 of file `qwt_plot_zoomer.cpp`.

6.65.3.10 void QwtPlotZoomer::setSelectionFlags (int *flags*) [virtual]

Set the selection flags

Parameters:

flags Or'd value of [QwtPicker::RectSelectionType](#) and [QwtPicker::SelectionMode](#). The default value is [QwtPicker::RectSelection](#) & [QwtPicker::ClickSelection](#).

See also:

[selectionFlags\(\)](#), [SelectionMode](#), [RectSelectionType](#), [SelectionMode](#)

Note:

[QwtPicker::RectSelection](#) will be auto added.

Reimplemented from [QwtPicker](#).

Definition at line 661 of file `qwt_plot_zoomer.cpp`.

References [QwtPicker::setSelectionFlags\(\)](#).

6.65.3.11 void QwtPlotZoomer::moveBy (double *dx*, double *dy*) [slot]

Move the current zoom rectangle.

Parameters:

dx X offset

dy Y offset

Note:

The changed rectangle is limited by the zoom base

Definition at line 494 of file `qwt_plot_zoomer.cpp`.

References [move\(\)](#).

6.65.3.12 void QwtPlotZoomer::move (double *x*, double *y*) [virtual, slot]

Move the the current zoom rectangle.

Parameters:

x X value

y Y value

See also:

QwtDoubleRect::move

Note:

The changed rectangle is limited by the zoom base

Definition at line 509 of file qwt_plot_zoomer.cpp.

References [rescale\(\)](#), [zoomBase\(\)](#), and [zoomRect\(\)](#).

Referenced by [moveBy\(\)](#).

6.65.3.13 void QwtPlotZoomer::zoom (const QwtDoubleRect & *rect*) [virtual, slot]

Zoom in.

Clears all rectangles above the current position of the zoom stack and pushes the intersection of [zoomRect\(\)](#) and the normalized rect on it.

Note:

If the maximal stack depth is reached, zoom is ignored.

The zoomed signal is emitted.

Definition at line 283 of file qwt_plot_zoomer.cpp.

References [rescale\(\)](#), [zoomed\(\)](#), and [zoomRect\(\)](#).

Referenced by [end\(\)](#), [setMaxStackDepth\(\)](#), [widgetKeyPressEvent\(\)](#), and [widgetMouseReleaseEvent\(\)](#).

6.65.3.14 void QwtPlotZoomer::zoom (int *offset*) [virtual, slot]

Zoom in or out.

Activate a rectangle on the zoom stack with an offset relative to the current position. Negative values of offset will zoom out, positive zoom in. A value of 0 zooms out to the zoom base.

Parameters:

offset Offset relative to the current position of the zoom stack.

Note:

The zoomed signal is emitted.

See also:

[zoomRectIndex\(\)](#)

Definition at line 320 of file qwt_plot_zoomer.cpp.

References [rescale\(\)](#), [zoomed\(\)](#), and [zoomRect\(\)](#).

6.65.3.15 void QwtPlotZoomer::zoomed (const [QwtDoubleRect](#) & *rect*) [signal]

A signal emitting the [zoomRect\(\)](#), when the plot has been zoomed in or out.

Parameters:

rect Current zoom rectangle.

Referenced by [zoom\(\)](#).

6.65.3.16 void QwtPlotZoomer::rescale () [protected, virtual]

Adjust the observed plot to [zoomRect\(\)](#)

Note:

Initiates [QwtPlot::replot](#)

Definition at line 385 of file `qwt_plot_zoomer.cpp`.

References [QwtPlot::autoReplot\(\)](#), [QwtPlot::axisScaleDiv\(\)](#), [QwtScaleDiv::hBound\(\)](#), [QwtScaleDiv::lBound\(\)](#), [QwtPlotPicker::plot\(\)](#), [QwtPlot::replot\(\)](#), [QwtPlotPicker::scaleRect\(\)](#), [QwtPlot::setAutoReplot\(\)](#), [QwtPlot::setAxisScale\(\)](#), [QwtPlotPicker::xAxis\(\)](#), and [QwtPlotPicker::yAxis\(\)](#).

Referenced by [move\(\)](#), [setZoomBase\(\)](#), and [zoom\(\)](#).

6.65.3.17 [QwtDoubleSize](#) QwtPlotZoomer::minZoomSize () const [protected, virtual]

Limit zooming by a minimum rectangle.

Returns:

[zoomBase\(\).width\(\)](#) / 10e4, [zoomBase\(\).height\(\)](#) / 10e4

Definition at line 573 of file `qwt_plot_zoomer.cpp`.

Referenced by [accept\(\)](#), [begin\(\)](#), and [end\(\)](#).

6.65.3.18 void QwtPlotZoomer::widgetMouseEvent (QMouseEvent * *me*) [protected, virtual]

Qt::MidButton zooms out one position on the zoom stack, Qt::RightButton to the zoom base.

Changes the current position on the stack, but doesn't pop any rectangle.

Note:

The mouse events can be changed, using [QwtEventPattern::setMousePattern](#): 2, 1

Reimplemented from [QwtPicker](#).

Definition at line 448 of file `qwt_plot_zoomer.cpp`.

References [QwtEventPattern::mouseMatch\(\)](#), [QwtPicker::widgetMouseEvent\(\)](#), and [zoom\(\)](#).

6.65.3.19 `void QwtPlotZoomer::widgetKeyPressEvent (QKeyEvent * ke) [protected, virtual]`

Qt::Key_Plus zooms out, Qt::Key_Minus zooms in one position on the zoom stack, Qt::Key_Escape zooms out to the zoom base.

Changes the current position on the stack, but doesn't pop any rectangle.

Note:

The keys codes can be changed, using [QwtEventPattern::setKeyPattern](#): 3, 4, 5

Reimplemented from [QwtPicker](#).

Definition at line 471 of file `qwt_plot_zoomer.cpp`.

References [QwtPicker::isActive\(\)](#), [QwtEventPattern::keyMatch\(\)](#), [QwtPicker::widgetKeyPressEvent\(\)](#), and [zoom\(\)](#).

6.65.3.20 `void QwtPlotZoomer::begin () [protected, virtual]`

Rejects selections, when the stack depth is too deep, or the zoomed rectangle is [minZoomSize\(\)](#).

See also:

[minZoomSize\(\)](#), [maxStackDepth\(\)](#)

Reimplemented from [QwtPicker](#).

Definition at line 587 of file `qwt_plot_zoomer.cpp`.

References [QwtPicker::begin\(\)](#), and [minZoomSize\(\)](#).

6.65.3.21 `bool QwtPlotZoomer::end (bool ok = true) [protected, virtual]`

Expand the selected rectangle to [minZoomSize\(\)](#) and zoom in if accepted.

See also:

[QwtPlotZoomer::accept\(a, QwtPlotZoomer::minZoomSize\(\)\)](#)

Reimplemented from [QwtPlotPicker](#).

Definition at line 617 of file `qwt_plot_zoomer.cpp`.

References [QwtPlotPicker::end\(\)](#), [QwtPlotPicker::invTransform\(\)](#), [minZoomSize\(\)](#), [QwtPlotPicker::plot\(\)](#), [QwtPicker::selection\(\)](#), [zoom\(\)](#), and [zoomRect\(\)](#).

6.65.3.22 `bool QwtPlotZoomer::accept (QwtPolygon & pa) const [protected, virtual]`

Check and correct a selected rectangle.

Reject rectangles with a hight or width < 2, otherwise expand the selected rectangle to a minimum size of 11x11 and accept it.

Returns:

true If rect is accepted, or has been changed to a accepted rectangle.

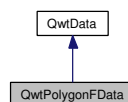
Reimplemented from [QwtPicker](#).

Definition at line 539 of file qwt_plot_zoomer.cpp.

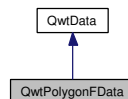
References [minZoomSize\(\)](#).

6.66 QwtPolygonFData Class Reference

Inheritance diagram for QwtPolygonFData:



Collaboration diagram for QwtPolygonFData:



6.66.1 Detailed Description

Data class containing a single `QwtArray<QwtDoublePoint>` object.

Definition at line 85 of file `qwt_data.h`.

Public Member Functions

- [QwtPolygonFData](#) (const `QPolygonF` &)
- [QwtPolygonFData & operator=](#) (const [QwtPolygonFData](#) &)
- virtual [QwtData](#) * [copy](#) () const
- virtual `size_t` [size](#) () const
- virtual double [x](#) (`size_t` i) const
- virtual double [y](#) (`size_t` i) const
- const `QPolygonF` & [data](#) () const

6.66.2 Constructor & Destructor Documentation

6.66.2.1 QwtPolygonFData::QwtPolygonFData (const `QPolygonF` & *polygon*)

Constructor

Parameters:

polygon Polygon data

See also:

[QwtPlotCurve::setData\(\)](#)

Definition at line 69 of file qwt_data.cpp.

Referenced by copy().

6.66.3 Member Function Documentation

6.66.3.1 [QwtPolygonFData](#) & [QwtPolygonFData::operator=](#) (const [QwtPolygonFData](#) &)

Assignment.

Definition at line 78 of file qwt_data.cpp.

References [QwtSpline::d_data](#), and [data\(\)](#).

6.66.3.2 [QwtData](#) * [QwtPolygonFData::copy](#) () const [virtual]

Returns:

Pointer to a copy (virtual copy constructor)

Implements [QwtData](#).

Definition at line 128 of file qwt_data.cpp.

References [QwtSpline::d_data](#), and [QwtPolygonFData\(\)](#).

6.66.3.3 [size_t](#) [QwtPolygonFData::size](#) () const [virtual]

Returns:

Size of the data set

Implements [QwtData](#).

Definition at line 89 of file qwt_data.cpp.

References [QwtSpline::d_data](#).

6.66.3.4 [double](#) [QwtPolygonFData::x](#) ([size_t](#) *i*) const [virtual]

Return the x value of data point *i*

Parameters:

i Index

Returns:

x X value of data point *i*

Implements [QwtData](#).

Definition at line 100 of file qwt_data.cpp.

References [QwtSpline::d_data](#).

6.66.3.5 `double QwtPolygonFData::y (size_t i) const` [virtual]

Return the y value of data point i

Parameters:

i Index

Returns:

y Y value of data point i

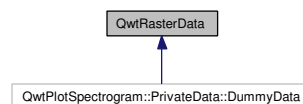
Implements [QwtData](#).

Definition at line 111 of file qwt_data.cpp.

References [QwtSpline::d_data](#).

6.67 QwtRasterData Class Reference

Inheritance diagram for QwtRasterData:

**6.67.1 Detailed Description**

[QwtRasterData](#) defines an interface to any type of raster data.

Definition at line 53 of file qwt_raster_data.h.

Public Types

- enum [ConrecAttribute](#) {
IgnoreAllVerticesOnLevel = 1,
IgnoreOutOfRange = 2 }
- typedef `QMap< double, QPolygonF >` [ContourLines](#)

Public Member Functions

- [QwtRasterData](#) ()
- [QwtRasterData](#) (const [QwtDoubleRect](#) &)
- virtual [~QwtRasterData](#) ()
- virtual [QwtRasterData * copy](#) () const=0
- virtual void [setBoundingRect](#) (const [QwtDoubleRect](#) &)
- [QwtDoubleRect boundingRect](#) () const
- virtual `QSize` [rasterHint](#) (const [QwtDoubleRect](#) &) const
- virtual void [initRaster](#) (const [QwtDoubleRect](#) &, const `QSize` &raster)
- virtual void [discardRaster](#) ()
- virtual double [value](#) (double x, double y) const=0

- virtual [QwtDoubleInterval range](#) () const=0
- virtual [ContourLines contourLines](#) (const [QwtDoubleRect](#) &rect, const QSize &raster, const QList<double> &levels, int flags) const

6.67.2 Member Function Documentation

6.67.2.1 virtual [QwtRasterData*](#) QwtRasterData::copy () const [pure virtual]

Clone the data.

Referenced by [QwtPlotSpectrogram::setData\(\)](#).

6.67.2.2 QSize QwtRasterData::rasterHint (const [QwtDoubleRect](#) &) const [virtual]

Find the raster of the data for an area.

The resolution is the number of horizontal and vertical pixels that the data can return for an area. An invalid resolution indicates that the data can return values for any detail level.

The resolution will limit the size of the image that is rendered from the data. F.e. this might be important when printing a spectrogram to a A0 printer with 600 dpi.

The default implementation returns an invalid resolution (size)

Parameters:

rect In most implementations the resolution of the data doesn't depend on the requested rectangle.

Returns:

Resolution, as number of horizontal and vertical pixels

Definition at line 258 of file [qwt_raster_data.cpp](#).

6.67.2.3 void QwtRasterData::initRaster (const [QwtDoubleRect](#) &, const QSize & raster) [virtual]

Initialize a raster.

Before the composition of an image [QwtPlotSpectrogram](#) calls [initRaster](#), announcing the area and its resolution that will be requested.

The default implementation does nothing, but for data sets that are stored in files, it might be good idea to reimplement [initRaster](#), where the data is resampled and loaded into memory.

Parameters:

rect Area of the raster

raster Number of horizontal and vertical pixels

See also:

[initRaster\(\)](#), [value\(\)](#)

Definition at line 222 of file [qwt_raster_data.cpp](#).

6.67.2.4 void QwtRasterData::discardRaster () [virtual]

Discard a raster.

After the composition of an image [QwtPlotSpectrogram](#) calls [discardRaster\(\)](#).

The default implementation does nothing, but if data has been loaded in [initRaster\(\)](#), it could be deleted now.

See also:

[initRaster\(\)](#), [value\(\)](#)

Definition at line 236 of file `qwt_raster_data.cpp`.

6.67.2.5 virtual double QwtRasterData::value (double x, double y) const [pure virtual]

Returns:

the value at a raster position

6.67.2.6 virtual QwtDoubleInterval QwtRasterData::range () const [pure virtual]

Returns:

the range of the values

6.67.2.7 QwtRasterData::ContourLines QwtRasterData::contourLines (const QwtDoubleRect & rect, const QSize & raster, const QList< double > & levels, int flags) const [virtual]

Calculate contour lines

An adaption of CONREC, a simple contouring algorithm. <http://local.wasp.uwa.edu.au/~pbourke/papers/c>

Definition at line 270 of file `qwt_raster_data.cpp`.

References [QwtDoubleInterval::contains\(\)](#), and [QwtDoubleInterval::isValid\(\)](#).

6.68 QwtRect Class Reference**6.68.1 Detailed Description**

Some extensions for `QRect`

Deprecated

Use [QwtClipper](#) instead.

Definition at line 22 of file `qwt_rect.h`.

Public Member Functions

- [QwtRect \(\)](#)
- [QwtRect \(const QRect &r\)](#)
- [QwtPolygon clip](#) (const [QwtPolygon](#) &) const

6.68.2 Constructor & Destructor Documentation

6.68.2.1 QwtRect::QwtRect ()

Constructor.

Definition at line 14 of file qwt_rect.cpp.

6.68.2.2 QwtRect::QwtRect (const QRect & r)

Copy constructor.

Definition at line 20 of file qwt_rect.cpp.

6.68.3 Member Function Documentation

6.68.3.1 QwtPolygon QwtRect::clip (const QwtPolygon &) const

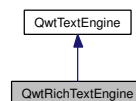
Sutherland-Hodgman polygon clipping.

Definition at line 26 of file qwt_rect.cpp.

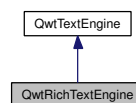
References `QwtClipper::clipPolygon()`.

6.69 QwtRichTextEngine Class Reference

Inheritance diagram for QwtRichTextEngine:



Collaboration diagram for QwtRichTextEngine:



6.69.1 Detailed Description

A text engine for Qt rich texts.

[QwtRichTextEngine](#) renders Qt rich texts using the classes of the Scribe framework of Qt.

Definition at line 150 of file qwt_text_engine.h.

Public Member Functions

- [QwtRichTextEngine](#) ()
- virtual int [heightForWidth](#) (const QFont &font, int flags, const QString &text, int width) const
- virtual QSize [textSize](#) (const QFont &font, int flags, const QString &text) const

- virtual void [draw](#) (QPainter *painter, const QRect &rect, int flags, const QString &text) const
- virtual bool [mightRender](#) (const QString &) const
- virtual void [textMargins](#) (const QFont &, const QString &, int &left, int &right, int &top, int &bottom) const

6.69.2 Constructor & Destructor Documentation

6.69.2.1 QwtRichTextEngine::QwtRichTextEngine ()

Constructor.

Definition at line 277 of file qwt_text_engine.cpp.

6.69.3 Member Function Documentation

6.69.3.1 int QwtRichTextEngine::heightForWidth (const QFont & *font*, int *flags*, const QString & *text*, int *width*) const [virtual]

Find the height for a given width

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

width Width

Returns:

Calculated height

Implements [QwtTextEngine](#).

Definition at line 291 of file qwt_text_engine.cpp.

6.69.3.2 QSize QwtRichTextEngine::textSize (const QFont & *font*, int *flags*, const QString & *text*) const [virtual]

Returns the size, that is needed to render text

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

Returns:

Calculated size

Implements [QwtTextEngine](#).

Definition at line 316 of file qwt_text_engine.cpp.

6.69.3.3 `void QwtRichTextEngine::draw (QPainter * painter, const QRect & rect, int flags, const QString & text) const` [virtual]

Draw the text in a clipping rectangle

Parameters:

painter Painter

rect Clipping rectangle

flags Bitwise OR of the flags like in for QPainter::drawText

text Text to be rendered

Implements [QwtTextEngine](#).

Definition at line 387 of file qwt_text_engine.cpp.

References [QwtPainter::drawSimpleRichText\(\)](#).

6.69.3.4 `bool QwtRichTextEngine::mightRender (const QString & text) const` [virtual]

Test if a string can be rendered by this text engine

Parameters:

text Text to be tested

Returns:

`QStyleSheet::mightBeRichText(text);`

Implements [QwtTextEngine](#).

Definition at line 413 of file qwt_text_engine.cpp.

6.69.3.5 `void QwtRichTextEngine::textMargins (const QFont &, const QString &, int & left, int & right, int & top, int & bottom) const` [virtual]

Return margins around the texts

Parameters:

left Return 0

right Return 0

top Return 0

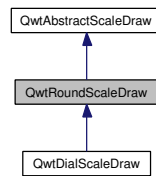
bottom Return 0

Implements [QwtTextEngine](#).

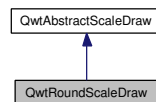
Definition at line 430 of file qwt_text_engine.cpp.

6.70 QwtRoundScaleDraw Class Reference

Inheritance diagram for QwtRoundScaleDraw:



Collaboration diagram for QwtRoundScaleDraw:



6.70.1 Detailed Description

A class for drawing round scales.

[QwtRoundScaleDraw](#) can be used to draw round scales. The circle segment can be adjusted by [QwtRoundScaleDraw::setAngleRange\(\)](#). The geometry of the scale can be specified with [QwtRoundScaleDraw::moveCenter\(\)](#) and [QwtRoundScaleDraw::setRadius\(\)](#).

After a scale division has been specified as a [QwtScaleDiv](#) object using [QwtAbstractScaleDraw::setScaleDiv\(const QwtScaleDiv &s\)](#), the scale can be drawn with the [QwtAbstractScaleDraw::draw\(\)](#) member.

Definition at line 32 of file `qwt_round_scale_draw.h`.

Public Member Functions

- [QwtRoundScaleDraw](#) ()
- [QwtRoundScaleDraw](#) (const [QwtRoundScaleDraw](#) &)
- virtual [~QwtRoundScaleDraw](#) ()
- [QwtRoundScaleDraw](#) & operator= (const [QwtRoundScaleDraw](#) &other)
- void [setRadius](#) (int radius)
- int [radius](#) () const
- void [moveCenter](#) (int x, int y)
- void [moveCenter](#) (const QPoint &)
- QPoint [center](#) () const
- void [setAngleRange](#) (double angle1, double angle2)
- virtual int [extent](#) (const QPen &, const QFont &) const

Protected Member Functions

- virtual void [drawTick](#) (QPainter *p, double val, int len) const
- virtual void [drawBackbone](#) (QPainter *p) const
- virtual void [drawLabel](#) (QPainter *p, double val) const

6.70.2 Constructor & Destructor Documentation

6.70.2.1 QwtRoundScaleDraw::QwtRoundScaleDraw ()

Constructor.

The range of the scale is initialized to [0, 100], The center is set to (50, 50) with a radius of 50. The angle range is set to [-135, 135].

Definition at line 46 of file qwt_round_scale_draw.cpp.

References `QwtAbstractScaleDraw::scaleMap()`, `QwtScaleMap::setPaintInterval()`, and `setRadius()`.

6.70.2.2 QwtRoundScaleDraw::QwtRoundScaleDraw (const [QwtRoundScaleDraw](#) &)

Copy constructor.

Definition at line 55 of file qwt_round_scale_draw.cpp.

References `d_data`.

6.70.2.3 QwtRoundScaleDraw::~QwtRoundScaleDraw () [virtual]

Destructor.

Definition at line 63 of file qwt_round_scale_draw.cpp.

6.70.3 Member Function Documentation

6.70.3.1 [QwtRoundScaleDraw](#) & QwtRoundScaleDraw::operator= (const [QwtRoundScaleDraw](#) & *other*)

Assignment operator.

Definition at line 69 of file qwt_round_scale_draw.cpp.

References `d_data`.

6.70.3.2 void QwtRoundScaleDraw::setRadius (int *radius*)

Change of radius the scale

Radius is the radius of the backbone without ticks and labels.

Parameters:

radius New Radius

See also:

[moveCenter](#)

Definition at line 84 of file qwt_round_scale_draw.cpp.

Referenced by `QwtRoundScaleDraw()`.

6.70.3.3 int QwtRoundScaleDraw::radius () const

Get the radius

Radius is the radius of the backbone without ticks and labels.

See also:

[setRadius\(\)](#), [extent\(\)](#)

Definition at line 96 of file qwt_round_scale_draw.cpp.

Referenced by [drawBackbone\(\)](#), [drawLabel\(\)](#), and [drawTick\(\)](#).

6.70.3.4 void QwtRoundScaleDraw::moveCenter (const QPoint & center)

Move the center of the scale draw, leaving the radius unchanged

Parameters:

center New center

See also:

[setRadius](#)

Definition at line 107 of file qwt_round_scale_draw.cpp.

6.70.3.5 QPoint QwtRoundScaleDraw::center () const

Get the center of the scale.

Definition at line 113 of file qwt_round_scale_draw.cpp.

6.70.3.6 void QwtRoundScaleDraw::setAngleRange (double angle1, double angle2)

Adjust the baseline circle segment for round scales.

The baseline will be drawn from min(angle1, angle2) to max(angle1, angle2). The default setting is [-135, 135]. An angle of 0 degrees corresponds to the 12 o'clock position, and positive angles count in a clockwise direction.

Parameters:

angle1

angle2 boundaries of the angle interval in degrees.

Warning:

- The angle range is limited to [-360, 360] degrees. Angles exceeding this range will be clipped.
- For angles more than 359 degrees above or below min(angle1, angle2), scale marks will not be drawn.
- If you need a counterclockwise scale, use [QwtScaleDiv::setRange](#)

Definition at line 135 of file qwt_round_scale_draw.cpp.

References [QwtAbstractScaleDraw::scaleMap\(\)](#), and [QwtScaleMap::setPaintInterval\(\)](#).

Referenced by [QwtKnob::setTotalAngle\(\)](#).

6.70.3.7 `int QwtRoundScaleDraw::extent (const QPen & pen, const QFont & font) const`
[virtual]

Calculate the extent of the scale

The extent is the distance between the baseline to the outermost pixel of the scale draw. `radius() + extent()` is an upper limit for the radius of the bounding circle.

Parameters:

pen Pen that is used for painting backbone and ticks

font Font used for painting the labels

See also:

[setMinimumExtent\(\)](#), [minimumExtent\(\)](#)

Warning:

The implemented algo is not too smart and calculates only an upper limit, that might be a few pixels too large

Implements [QwtAbstractScaleDraw](#).

Definition at line 267 of file `qwt_round_scale_draw.cpp`.

References [QwtScaleDiv::contains\(\)](#), [QwtAbstractScaleDraw::hasComponent\(\)](#), [QwtText::isEmpty\(\)](#), [QwtAbstractScaleDraw::label\(\)](#), [QwtAbstractScaleDraw::majTickLength\(\)](#), [QwtAbstractScaleDraw::map\(\)](#), [QwtAbstractScaleDraw::minimumExtent\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), [QwtAbstractScaleDraw::spacing\(\)](#), [QwtText::textSize\(\)](#), [QwtAbstractScaleDraw::tickLabel\(\)](#), [QwtScaleDiv::ticks\(\)](#), and [QwtScaleMap::transform\(\)](#).

Referenced by [QwtKnob::minimumSizeHint\(\)](#).

6.70.3.8 `void QwtRoundScaleDraw::drawTick (QPainter * painter, double value, int len) const`
[protected, virtual]

Draw a tick

Parameters:

painter Painter

value Value of the tick

len Length of the tick

See also:

[drawBackbone\(\)](#), [drawLabel\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 205 of file `qwt_round_scale_draw.cpp`.

References [QwtPainter::drawLine\(\)](#), [QwtAbstractScaleDraw::map\(\)](#), [radius\(\)](#), and [QwtScaleMap::transform\(\)](#).

6.70.3.9 `void QwtRoundScaleDraw::drawBackbone (QPainter * painter) const` [protected, virtual]

Draws the baseline of the scale

Parameters:

painter Painter

See also:

[drawTick\(\)](#), [drawLabel\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 239 of file `qwt_round_scale_draw.cpp`.

References [QwtAbstractScaleDraw::map\(\)](#), and [radius\(\)](#).

6.70.3.10 `void QwtRoundScaleDraw::drawLabel (QPainter * painter, double value) const` [protected, virtual]

Draws the label for a major scale tick

Parameters:

painter Painter

value Value

See also:

[drawTick\(\)](#), [drawBackbone\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 160 of file `qwt_round_scale_draw.cpp`.

References [QwtText::draw\(\)](#), [QwtAbstractScaleDraw::hasComponent\(\)](#), [QwtText::isEmpty\(\)](#), [QwtAbstractScaleDraw::label\(\)](#), [QwtAbstractScaleDraw::majTickLength\(\)](#), [QwtAbstractScaleDraw::map\(\)](#), [radius\(\)](#), [QwtAbstractScaleDraw::spacing\(\)](#), [QwtText::textSize\(\)](#), [QwtAbstractScaleDraw::tickLabel\(\)](#), and [QwtScaleMap::transform\(\)](#).

6.71 QwtScaleArithmetic Class Reference

6.71.1 Detailed Description

Arithmetic including a tolerance.

Definition at line 22 of file `qwt_scale_engine.h`.

Static Public Member Functions

- static int [compareEps](#) (double value1, double value2, double intervalSize)
- static double [ceilEps](#) (double value, double intervalSize)
- static double [floorEps](#) (double value, double intervalSize)
- static double [divideEps](#) (double interval, double steps)
- static double [ceil125](#) (double x)
- static double [floor125](#) (double x)

6.71.2 Member Function Documentation

6.71.2.1 `int QwtScaleArithmetic::compareEps (double value1, double value2, double intervalSize)` `[static]`

Compare 2 values, relative to an interval.

Values are "equal", when : $\cdot value2 - value1 \leq \text{abs}(intervalSize * 10e^{-6})$

Parameters:

value1 First value to compare

value2 Second value to compare

intervalSize interval size

Returns:

0: if equal, -1: if $value2 > value1$, 1: if $value1 > value2$

Definition at line 28 of file `qwt_scale_engine.cpp`.

Referenced by `QwtScaleEngine::contains()`.

6.71.2.2 `double QwtScaleArithmetic::ceilEps (double value, double intervalSize)` `[static]`

Ceil a value, relative to an interval

Parameters:

value Value to ceil

intervalSize Interval size

See also:

[floorEps](#)

Definition at line 50 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLinearScaleEngine::align()`.

6.71.2.3 `double QwtScaleArithmetic::floorEps (double value, double intervalSize)` `[static]`

Floor a value, relative to an interval

Parameters:

value Value to floor

intervalSize Interval size

See also:

[ceilEps](#)

Definition at line 67 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLinearScaleEngine::align()`.

6.71.2.4 double QwtScaleArithmetic::ceil125 (double *x*) [static]

Find the smallest value out of $\{1,2,5\} \cdot 10^n$ with an integer number *n* which is greater than or equal to *x*.

Parameters:

x Input value

Definition at line 98 of file qwt_scale_engine.cpp.

Referenced by QwtScaleEngine::divideInterval().

6.71.2.5 double QwtScaleArithmetic::floor125 (double *x*) [static]

Find the largest value out of $\{1,2,5\} \cdot 10^n$ with an integer number *n* which is smaller than or equal to *x*.

Parameters:

x Input value

Definition at line 126 of file qwt_scale_engine.cpp.

6.72 QwtScaleDiv Class Reference

6.72.1 Detailed Description

A class representing a scale division.

A scale division consists of its limits and 3 list of tick values qualified as major, medium and minor ticks.

In most cases scale divisions are calculated by a [QwtScaleEngine](#).

See also:

[QwtScaleEngine::subDivideInto](#), [QwtScaleEngine::subDivide](#)

Definition at line 30 of file qwt_scale_div.h.

Public Types

- enum [TickType](#) {
 NoTick = -1,
 MinorTick,
 MediumTick,
 MajorTick,
 NTickTypes }

Public Member Functions

- [QwtScaleDiv](#) ()
- [QwtScaleDiv](#) (const [QwtDoubleInterval](#) &, QwtValueList[NTickTypes])
- [QwtScaleDiv](#) (double lBound, double rBound, QwtValueList[NTickTypes])

- int `operator==` (const [QwtScaleDiv](#) &s) const
- int `operator!=` (const [QwtScaleDiv](#) &s) const
- void `setInterval` (double lBound, double rBound)
- void `setInterval` (const [QwtDoubleInterval](#) &)
- [QwtDoubleInterval](#) `interval` () const
- double `lBound` () const
- double `hBound` () const
- double `range` () const
- bool `contains` (double v) const
- void `setTicks` (int type, const [QwtValueList](#) &)
- const [QwtValueList](#) & `ticks` (int type) const
- void `invalidate` ()
- bool `isValid` () const
- void `invert` ()

6.72.2 Constructor & Destructor Documentation

6.72.2.1 [QwtScaleDiv::QwtScaleDiv](#) () [explicit]

Construct an invalid [QwtScaleDiv](#) instance.

Definition at line 15 of file `qwt_scale_div.cpp`.

6.72.2.2 [QwtScaleDiv::QwtScaleDiv](#) (const [QwtDoubleInterval](#) & *interval*, [QwtValueList](#) *ticks*[[NTickTypes](#)]) [explicit]

Construct [QwtScaleDiv](#) instance.

Parameters:

interval Interval

ticks List of major, medium and minor ticks

Definition at line 28 of file `qwt_scale_div.cpp`.

References `ticks()`.

6.72.2.3 [QwtScaleDiv::QwtScaleDiv](#) (double *lBound*, double *hBound*, [QwtValueList](#) *ticks*[[NTickTypes](#)]) [explicit]

Construct [QwtScaleDiv](#) instance.

Parameters:

lBound First interval limit

hBound Second interval limit

ticks List of major, medium and minor ticks

Definition at line 46 of file `qwt_scale_div.cpp`.

References `ticks()`.

6.72.3 Member Function Documentation

6.72.3.1 `int QwtScaleDiv::operator==(const QwtScaleDiv & other) const`

Equality operator.

Returns:

true if this instance is equal to other

Definition at line 70 of file qwt_scale_div.cpp.

References d_hBound, d_isValid, d_lBound, and d_ticks.

6.72.3.2 `int QwtScaleDiv::operator!=(const QwtScaleDiv & s) const`

Inequality.

Returns:

true if this instance is not equal to s

Definition at line 92 of file qwt_scale_div.cpp.

6.72.3.3 `void QwtScaleDiv::setInterval (double lBound, double hBound) [inline]`

Change the interval left bound right bound

Definition at line 84 of file qwt_scale_div.h.

Referenced by setInterval().

6.72.3.4 `void QwtScaleDiv::setInterval (const QwtDoubleInterval & interval)`

Change the interval Interval

Definition at line 61 of file qwt_scale_div.cpp.

References interval(), QwtDoubleInterval::maxValue(), QwtDoubleInterval::minValue(), and setInterval().

6.72.3.5 `QwtDoubleInterval QwtScaleDiv::interval () const [inline]`

Returns:

lBound -> hBound

Definition at line 93 of file qwt_scale_div.h.

Referenced by setInterval().

6.72.3.6 `double QwtScaleDiv::lBound () const [inline]`

Returns:

left bound

See also:

[QwtScaleDiv::hBound](#)

Definition at line 102 of file qwt_scale_div.h.

Referenced by `QwtPlot::canvasMap()`, `QwtPlotPanner::moveCanvas()`, `QwtPlot::print()`, `QwtPlotZoomer::rescale()`, `QwtPlotMagnifier::rescale()`, `QwtPlotPicker::scaleRect()`, and `QwtAbstractScaleDraw::setScaleDiv()`.

6.72.3.7 double QwtScaleDiv::hBound () const [inline]

Returns:

right bound

See also:

[QwtScaleDiv::lBound](#)

Definition at line 111 of file qwt_scale_div.h.

Referenced by `QwtPlot::canvasMap()`, `QwtPlotPanner::moveCanvas()`, `QwtPlot::print()`, `QwtPlotZoomer::rescale()`, and `QwtAbstractScaleDraw::setScaleDiv()`.

6.72.3.8 double QwtScaleDiv::range () const [inline]

Returns:

`hBound() - lBound()`

Definition at line 119 of file qwt_scale_div.h.

Referenced by `QwtPlotPicker::scaleRect()`.

6.72.3.9 void QwtScaleDiv::setTicks (int type, const QwtValueList & ticks)

Assign ticks

Parameters:

type MinorTick, MediumTick or MajorTick

ticks Values of the tick positions

Definition at line 149 of file qwt_scale_div.cpp.

6.72.3.10 const QwtValueList & QwtScaleDiv::ticks (int type) const

Return a list of ticks

Parameters:

type MinorTick, MediumTick or MajorTick

Definition at line 160 of file qwt_scale_div.cpp.

Referenced by `QwtRoundScaleDraw::extent()`, `QwtScaleDraw::getBorderDistHint()`, `invert()`, `QwtScaleDraw::maxLabelHeight()`, `QwtScaleDraw::maxLabelWidth()`, `QwtScaleDraw::minLabelDist()`, `QwtScaleDraw::minLength()`, `QwtScaleDiv()`, and `QwtPlot::sizeHint()`.

6.72.3.11 void QwtScaleDiv::invalidate ()

Invalidate the scale division.

Definition at line 98 of file qwt_scale_div.cpp.

6.72.3.12 bool QwtScaleDiv::isValid () const

Check if the scale division is valid.

Definition at line 110 of file qwt_scale_div.cpp.

6.72.3.13 void QwtScaleDiv::invert ()

Invert the scale division.

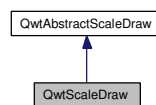
Definition at line 127 of file qwt_scale_div.cpp.

References ticks().

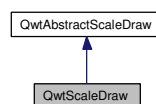
Referenced by QwtLog10ScaleEngine::divideScale().

6.73 QwtScaleDraw Class Reference

Inheritance diagram for QwtScaleDraw:



Collaboration diagram for QwtScaleDraw:

**6.73.1 Detailed Description**

A class for drawing scales.

[QwtScaleDraw](#) can be used to draw linear or logarithmic scales. A scale has a position, an alignment and a length, which can be specified. The labels can be rotated and aligned to the ticks using [setLabelRotation\(\)](#) and [setLabelAlignment\(\)](#).

After a scale division has been specified as a [QwtScaleDiv](#) object using [QwtAbstractScaleDraw::setScaleDiv\(const QwtScaleDiv &s\)](#), the scale can be drawn with the [QwtAbstractScaleDraw::draw\(\)](#) member.

Definition at line 30 of file qwt_scale_draw.h.

Public Types

- enum [Alignment](#) {


```

    BottomScale,
    TopScale,
    LeftScale,
    RightScale }

```

Public Member Functions

- [QwtScaleDraw](#) ()
- [QwtScaleDraw](#) (const [QwtScaleDraw](#) &)
- virtual [~QwtScaleDraw](#) ()
- [QwtScaleDraw](#) & [operator=](#) (const [QwtScaleDraw](#) &other)
- void [getBorderDistHint](#) (const QFont &, int &start, int &end) const
- int [minLabelDist](#) (const QFont &) const
- int [minLength](#) (const QPen &, const QFont &) const
- virtual int [extent](#) (const QPen &, const QFont &) const
- void [move](#) (int x, int y)
- void [move](#) (const QPoint &)
- void [setLength](#) (int length)
- [Alignment](#) [alignment](#) () const
- void [setAlignment](#) ([Alignment](#))
- Qt::Orientation [orientation](#) () const
- QPoint [pos](#) () const
- int [length](#) () const
- void [setLabelAlignment](#) (Qt::Alignment)
- Qt::Alignment [labelAlignment](#) () const
- void [setLabelRotation](#) (double rotation)
- double [labelRotation](#) () const
- int [maxLabelHeight](#) (const QFont &) const
- int [maxLabelWidth](#) (const QFont &) const
- QPoint [labelPosition](#) (double val) const
- QRect [labelRect](#) (const QFont &, double val) const
- QSize [labelSize](#) (const QFont &, double val) const
- QRect [boundingLabelRect](#) (const QFont &, double val) const

Protected Member Functions

- QMatrix [labelMatrix](#) (const QPoint &, const QSize &) const
- virtual void [drawTick](#) (QPainter *p, double val, int len) const
- virtual void [drawBackbone](#) (QPainter *p) const
- virtual void [drawLabel](#) (QPainter *p, double val) const

6.73.2 Member Enumeration Documentation

6.73.2.1 enum [QwtScaleDraw::Alignment](#)

Alignment of the scale draw

See also:

[setAlignment\(\)](#), [alignment\(\)](#)

Definition at line 37 of file `qwt_scale_draw.h`.

6.73.3 Constructor & Destructor Documentation

6.73.3.1 QwtScaleDraw::QwtScaleDraw ()

Constructor.

The range of the scale is initialized to [0, 100], The position is at (0, 0) with a length of 100. The orientation is QwtAbstractScaleDraw::Bottom.

Definition at line 60 of file qwt_scale_draw.cpp.

References `setLength()`.

6.73.3.2 QwtScaleDraw::QwtScaleDraw (const QwtScaleDraw &)

Copy constructor.

Definition at line 67 of file qwt_scale_draw.cpp.

References `d_data`.

6.73.3.3 QwtScaleDraw::~QwtScaleDraw () [virtual]

Destructor.

Definition at line 74 of file qwt_scale_draw.cpp.

6.73.4 Member Function Documentation

6.73.4.1 QwtScaleDraw & QwtScaleDraw::operator= (const QwtScaleDraw & other)

Assignment operator.

Definition at line 80 of file qwt_scale_draw.cpp.

References `d_data`.

6.73.4.2 void QwtScaleDraw::getBorderDistHint (const QFont & font, int & start, int & end) const

Determine the minimum border distance.

This member function returns the minimum space needed to draw the mark labels at the scale's endpoints.

Parameters:

font Font

start Start border distance

end End border distance

Definition at line 139 of file qwt_scale_draw.cpp.

References `QwtAbstractScaleDraw::hasComponent()`, `labelRect()`, `QwtAbstractScaleDraw::map()`, `orientation()`, `QwtAbstractScaleDraw::scaleDiv()`, and `QwtScaleDiv::ticks()`.

Referenced by `QwtSlider::layoutSlider()`, `QwtThermo::layoutThermo()`, `QwtSlider::minimumSizeHint()`, and `minLength()`.

6.73.4.3 int QwtScaleDraw::minLabelDist (const QFont & *font*) const

Determine the minimum distance between two labels, that is necessary that the texts don't overlap.

Parameters:

font Font

Returns:

The maximum width of a label

See also:

[getBorderDistHint\(\)](#)

Definition at line 192 of file qwt_scale_draw.cpp.

References [QwtAbstractScaleDraw::hasComponent\(\)](#), [labelRect\(\)](#), [labelRotation\(\)](#), [orientation\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), and [QwtScaleDiv::ticks\(\)](#).

Referenced by [minLength\(\)](#).

6.73.4.4 int QwtScaleDraw::minLength (const QPen & *pen*, const QFont & *font*) const

Calculate the minimum length that is needed to draw the scale

Parameters:

pen Pen that is used for painting backbone and ticks

font Font used for painting the labels

See also:

[extent\(\)](#)

Definition at line 318 of file qwt_scale_draw.cpp.

References [getBorderDistHint\(\)](#), [QwtAbstractScaleDraw::hasComponent\(\)](#), [minLabelDist\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), and [QwtScaleDiv::ticks\(\)](#).

Referenced by [QwtThermo::minimumSizeHint\(\)](#), and [QwtSlider::minimumSizeHint\(\)](#).

6.73.4.5 int QwtScaleDraw::extent (const QPen & *pen*, const QFont & *font*) const [virtual]

Calculate the width/height that is needed for a vertical/horizontal scale.

The extent is calculated from the pen width of the backbone, the major tick length, the spacing and the maximum width/height of the labels.

Parameters:

pen Pen that is used for painting backbone and ticks

font Font used for painting the labels

See also:

[minLength\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 280 of file `qwt_scale_draw.cpp`.

References `QwtAbstractScaleDraw::hasComponent()`, `QwtAbstractScaleDraw::majTickLength()`, `maxLabelHeight()`, `maxLabelWidth()`, `QwtAbstractScaleDraw::minimumExtent()`, `orientation()`, and `QwtAbstractScaleDraw::spacing()`.

Referenced by `QwtThermo::minimumSizeHint()`, and `QwtSlider::minimumSizeHint()`.

6.73.4.6 void QwtScaleDraw::move (const QPoint & pos)

Move the position of the scale.

The meaning of the parameter `pos` depends on the alignment:

QwtScaleDraw::LeftScale The origin is the topmost point of the backbone. The backbone is a vertical line. Scale marks and labels are drawn at the left of the backbone.

QwtScaleDraw::RightScale The origin is the topmost point of the backbone. The backbone is a vertical line. Scale marks and labels are drawn at the right of the backbone.

QwtScaleDraw::TopScale The origin is the leftmost point of the backbone. The backbone is a horizontal line. Scale marks and labels are drawn above the backbone.

QwtScaleDraw::BottomScale The origin is the leftmost point of the backbone. The backbone is a horizontal line. Scale marks and labels are drawn below the backbone.

Parameters:

pos Origin of the scale

See also:

[pos\(\)](#), [setLength\(\)](#)

Definition at line 566 of file `qwt_scale_draw.cpp`.

6.73.4.7 void QwtScaleDraw::setLength (int length)

Set the length of the backbone.

The length doesn't include the space needed for overlapping labels.

See also:

[move\(\)](#), [minLabelDist\(\)](#)

Definition at line 589 of file `qwt_scale_draw.cpp`.

Referenced by `QwtPlotScaleItem::draw()`, `QwtSlider::layoutSlider()`, `QwtThermo::layoutThermo()`, `QwtPlot::printScale()`, and `QwtScaleDraw()`.

6.73.4.8 QwtScaleDraw::Alignment QwtScaleDraw::alignment () const

Return alignment of the scale

See also:

[setAlignment\(\)](#)

Definition at line 91 of file qwt_scale_draw.cpp.

Referenced by QwtScaleWidget::alignment(), QwtPlotScaleItem::draw(), drawBackbone(), drawTick(), labelMatrix(), labelPosition(), QwtPlotScaleItem::setAlignment(), and QwtSlider::setScaleDraw().

6.73.4.9 void QwtScaleDraw::setAlignment ([Alignment align](#))

Set the alignment of the scale

The default alignment is QwtScaleDraw::BottomScale

See also:

[alignment\(\)](#)

Definition at line 102 of file qwt_scale_draw.cpp.

Referenced by QwtThermo::layoutThermo(), QwtPlotScaleItem::setAlignment(), QwtSlider::setScaleDraw(), QwtScaleWidget::setScaleDraw(), and QwtSlider::setScalePosition().

6.73.4.10 Qt::Orientation QwtScaleDraw::orientation () const

Return the orientation

TopScale, BottomScale are horizontal (Qt::Horizontal) scales, LeftScale, RightScale are vertical (Qt::Vertical) scales.

See also:

[alignment\(\)](#)

Definition at line 115 of file qwt_scale_draw.cpp.

Referenced by QwtPlotScaleItem::draw(), QwtScaleWidget::drawColorBar(), drawTick(), extent(), getBorderDistHint(), minLabelDist(), QwtPlot::printScale(), and QwtPlotScaleItem::updateScaleDiv().

6.73.4.11 QPoint QwtScaleDraw::pos () const

Returns:

Origin of the scale

See also:

[move\(\)](#), [length\(\)](#)

Definition at line 576 of file qwt_scale_draw.cpp.

Referenced by boundingLabelRect(), drawBackbone(), drawLabel(), drawTick(), labelRect(), and QwtPlot::printScale().

6.73.4.12 int QwtScaleDraw::length () const

Returns:

the length of the backbone

See also:

[setLength\(\)](#), [pos\(\)](#)

Definition at line 604 of file `qwt_scale_draw.cpp`.

Referenced by `QwtPlot::printScale()`.

6.73.4.13 void QwtScaleDraw::setLabelAlignment (Qt::Alignment alignment)

Change the label flags.

Labels are aligned to the point ticklength + spacing away from the backbone.

The alignment is relative to the orientation of the label text. In case of an flags of 0 the label will be aligned depending on the orientation of the scale:

QwtScaleDraw::TopScale: Qt::AlignHCenter | Qt::AlignTop

QwtScaleDraw::BottomScale: Qt::AlignHCenter | Qt::AlignBottom

QwtScaleDraw::LeftScale: Qt::AlignLeft | Qt::AlignVCenter

QwtScaleDraw::RightScale: Qt::AlignRight | Qt::AlignVCenter

Changing the alignment is often necessary for rotated labels.

Parameters:

alignment Or'd Qt::AlignmentFlags <see `qnamespace.h`>

See also:

[setLabelRotation\(\)](#), [labelRotation\(\)](#), [labelAlignment\(\)](#)

Warning:

The various alignments might be confusing. The alignment of the label is not the alignment of the scale and is not the alignment of the flags (`QwtText::flags()`) returned from [QwtAbstractScaleDraw::label\(\)](#).

Definition at line 851 of file `qwt_scale_draw.cpp`.

6.73.4.14 Qt::Alignment QwtScaleDraw::labelAlignment () const

Returns:

the label flags

See also:

[setLabelAlignment\(\)](#), [labelRotation\(\)](#)

Definition at line 864 of file `qwt_scale_draw.cpp`.

Referenced by `labelMatrix()`.

6.73.4.15 void QwtScaleDraw::setLabelRotation (double rotation)

Rotate all labels.

When changing the rotation, it might be necessary to adjust the label flags too. Finding a useful combination is often the result of try and error.

Parameters:

rotation Angle in degrees. When changing the label rotation, the label flags often needs to be adjusted too.

See also:

[setLabelAlignment\(\)](#), [labelRotation\(\)](#), [labelAlignment\(\)](#).

Definition at line 809 of file qwt_scale_draw.cpp.

6.73.4.16 double QwtScaleDraw::labelRotation () const**Returns:**

the label rotation

See also:

[setLabelRotation\(\)](#), [labelAlignment\(\)](#)

Definition at line 818 of file qwt_scale_draw.cpp.

Referenced by [labelMatrix\(\)](#), and [minLabelDist\(\)](#).

6.73.4.17 int QwtScaleDraw::maxLabelHeight (const QFont & font) const**Parameters:**

font Font

Returns:

the maximum height of a label

Definition at line 897 of file qwt_scale_draw.cpp.

References [labelSize\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), and [QwtScaleDiv::ticks\(\)](#).

Referenced by [extent\(\)](#).

6.73.4.18 int QwtScaleDraw::maxLabelWidth (const QFont & font) const**Parameters:**

font Font

Returns:

the maximum width of a label

Definition at line 874 of file qwt_scale_draw.cpp.

References [labelSize\(\)](#), [QwtAbstractScaleDraw::scaleDiv\(\)](#), and [QwtScaleDiv::ticks\(\)](#).

Referenced by [extent\(\)](#).

6.73.4.19 QPoint QwtScaleDraw::labelPosition (double *value*) const

Find the position, where to paint a label

The position has a distance of `majTickLength() + spacing() + 1` from the backbone. The direction depends on the `alignment()`

Parameters:

value Value

Definition at line 356 of file `qwt_scale_draw.cpp`.

References `alignment()`, `QwtAbstractScaleDraw::hasComponent()`, `QwtAbstractScaleDraw::majTickLength()`, `QwtAbstractScaleDraw::map()`, `QwtAbstractScaleDraw::spacing()`, and `QwtScaleMap::transform()`.

Referenced by `boundingLabelRect()`, `drawLabel()`, and `labelRect()`.

6.73.4.20 QRect QwtScaleDraw::labelRect (const QFont & *font*, double *value*) const

Find the bounding rect for the label. The coordinates of the rect are relative to `spacing + ticklength` from the backbone in direction of the tick.

Parameters:

font Font used for painting

value Value

Definition at line 747 of file `qwt_scale_draw.cpp`.

References `QwtText::isEmpty()`, `labelMatrix()`, `labelPosition()`, `labelSize()`, `pos()`, `QwtText::textSize()`, `QwtAbstractScaleDraw::tickLabel()`, and `QwtMetricsMap::translate()`.

Referenced by `getBorderDistHint()`, `labelSize()`, and `minLabelDist()`.

6.73.4.21 QSize QwtScaleDraw::labelSize (const QFont & *font*, double *value*) const

Calculate the size that is needed to draw a label

Parameters:

font Label font

value Value

Definition at line 791 of file `qwt_scale_draw.cpp`.

References `labelRect()`.

Referenced by `boundingLabelRect()`, `drawLabel()`, `labelRect()`, `maxLabelHeight()`, and `maxLabelWidth()`.

6.73.4.22 QRect QwtScaleDraw::boundingLabelRect (const QFont & *font*, double *value*) const

Find the bounding rect for the label. The coordinates of the rect are absolute coordinates (calculated from `pos()`). in direction of the tick.

Parameters:

font Font used for painting

value Value

See also:

[labelRect\(\)](#)

Definition at line 652 of file `qwt_scale_draw.cpp`.

References `QwtText::isEmpty()`, `labelMatrix()`, `labelPosition()`, `labelSize()`, `pos()`, `QwtText::textSize()`, and `QwtAbstractScaleDraw::tickLabel()`.

6.73.4.23 QMatrix QwtScaleDraw::labelMatrix (const QPoint & *pos*, const QSize & *size*) const [protected]

Calculate the matrix that is needed to paint a label depending on its alignment and rotation.

Parameters:

pos Position where to paint the label

size Size of the label

See also:

[setLabelAlignment\(\)](#), [setLabelRotation\(\)](#)

Definition at line 676 of file `qwt_scale_draw.cpp`.

References `alignment()`, `labelAlignment()`, and `labelRotation()`.

Referenced by `boundingLabelRect()`, `drawLabel()`, and `labelRect()`.

6.73.4.24 void QwtScaleDraw::drawTick (QPainter * *painter*, double *value*, int *len*) const [protected, virtual]

Draw a tick

Parameters:

painter Painter

value Value of the tick

len Length of the tick

See also:

[drawBackbone\(\)](#), [drawLabel\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 406 of file `qwt_scale_draw.cpp`.

References `alignment()`, `QwtPainter::drawLine()`, `QwtMetricsMap::isIdentity()`, `QwtMetricsMap::layoutToDevice()`, `QwtMetricsMap::layoutToDeviceX()`, `QwtMetricsMap::layoutToDeviceY()`, `QwtAbstractScaleDraw::map()`, `QwtPainter::metricsMap()`, `orientation()`, `pos()`, `QwtPainter::resetMetricsMap()`, `QwtAbstractScaleDraw::scaleMap()`, and `QwtPainter::setMetricsMap()`.

6.73.4.25 `void QwtScaleDraw::drawBackbone (QPainter * painter) const` [protected, virtual]

Draws the baseline of the scale

Parameters:

painter Painter

See also:

[drawTick\(\)](#), [drawLabel\(\)](#)

Implements [QwtAbstractScaleDraw](#).

Definition at line 507 of file `qwt_scale_draw.cpp`.

References [alignment\(\)](#), [QwtPainter::drawLine\(\)](#), and [pos\(\)](#).

6.73.4.26 `void QwtScaleDraw::drawLabel (QPainter * painter, double value) const` [protected, virtual]

Draws the label for a major scale tick

Parameters:

painter Painter

value Value

See also:

[drawTick\(\)](#), [drawBackbone\(\)](#), [boundingLabelRect\(\)](#)

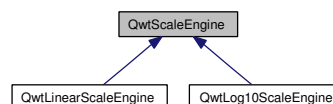
Implements [QwtAbstractScaleDraw](#).

Definition at line 617 of file `qwt_scale_draw.cpp`.

References [QwtText::draw\(\)](#), [QwtText::isEmpty\(\)](#), [labelMatrix\(\)](#), [labelPosition\(\)](#), [labelSize\(\)](#), [pos\(\)](#), [QwtText::textSize\(\)](#), and [QwtAbstractScaleDraw::tickLabel\(\)](#).

6.74 QwtScaleEngine Class Reference

Inheritance diagram for QwtScaleEngine:



6.74.1 Detailed Description

Base class for scale engines.

A scale engine tries to find "reasonable" ranges and step sizes for scales.

The layout of the scale can be varied with [setAttribute\(\)](#).

Qwt offers implementations for logarithmic (log10) and linear scales. Contributions for other types of scale engines (date/time, log2 ...) are welcome.

Definition at line 50 of file qwt_scale_engine.h.

Public Types

- enum [Attribute](#) {
 NoAttribute = 0,
 IncludeReference = 1,
 Symmetric = 2,
 Floating = 4,
 Inverted = 8 }

Public Member Functions

- [QwtScaleEngine](#) ()
- virtual [~QwtScaleEngine](#) ()
- void [setAttribute](#) ([Attribute](#), bool on=true)
- bool [testAttribute](#) ([Attribute](#)) const
- void [setAttributes](#) (int)
- int [attributes](#) () const
- void [setReference](#) (double reference)
- double [reference](#) () const
- void [setMargins](#) (double m1, double m2)
- double [loMargin](#) () const
- double [hiMargin](#) () const
- virtual void [autoScale](#) (int maxNumSteps, double &x1, double &x2, double &stepSize) const =0
- virtual [QwtScaleDiv](#) [divideScale](#) (double x1, double x2, int maxMajSteps, int maxMinSteps, double stepSize=0.0) const=0
- virtual [QwtScaleTransformation](#) * [transformation](#) () const=0

Protected Member Functions

- bool [contains](#) (const [QwtDoubleInterval](#) &, double val) const
- [QwtValueList](#) [strip](#) (const [QwtValueList](#) &, const [QwtDoubleInterval](#) &) const
- double [divideInterval](#) (double interval, int numSteps) const
- [QwtDoubleInterval](#) [buildInterval](#) (double v) const

6.74.2 Member Enumeration Documentation

6.74.2.1 enum [QwtScaleEngine::Attribute](#)

see [QwtScaleEngine::setAttribute](#), [testAttribute](#)

Definition at line 54 of file qwt_scale_engine.h.

6.74.3 Constructor & Destructor Documentation

6.74.3.1 QwtScaleEngine::QwtScaleEngine () [explicit]

Ctor.

Definition at line 169 of file qwt_scale_engine.cpp.

6.74.3.2 QwtScaleEngine::~QwtScaleEngine () [virtual]

Dtor.

Definition at line 176 of file qwt_scale_engine.cpp.

6.74.4 Member Function Documentation

6.74.4.1 void QwtScaleEngine::setAttribute ([Attribute](#) attribute, bool on = true)

Change a scale attribute

Parameters:

attribute Attribute to change

on On/Off

The behaviour of the scale engine can be changed with the following attributes:

QwtScaleEngine::IncludeReference Build a scale which includes the reference value.

QwtScaleEngine::Symmetric Build a scale which is symmetric to the reference value.

QwtScaleEngine::Floating The endpoints of the scale are supposed to be equal the outmost included values plus the specified margins (see [setMargins\(\)](#)). If this attribute is not* set, the endpoints of the scale will be integer multiples of the step size.

QwtScaleEngine::Inverted Turn the scale upside down.

See also:

[QwtScaleEngine::testAttribute\(\)](#)

Definition at line 339 of file qwt_scale_engine.cpp.

6.74.4.2 bool QwtScaleEngine::testAttribute ([Attribute](#) attribute) const

Check if a attribute is set.

Parameters:

attribute Attribute to be tested

See also:

[QwtScaleEngine::setAttribute\(\)](#) for a description of the possible options.

Definition at line 353 of file qwt_scale_engine.cpp.

Referenced by [QwtLog10ScaleEngine::autoScale\(\)](#), and [QwtLinearScaleEngine::autoScale\(\)](#).

6.74.4.3 void QwtScaleEngine::setAttributes (int *attributes*)

Change the scale attribute

Parameters:

attributes Set scale attributes

See also:

[QwtScaleEngine::attributes\(\)](#)

Definition at line 364 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::divideScale()`.

6.74.4.4 int QwtScaleEngine::attributes () const

Return the scale attributes

Definition at line 372 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::divideScale()`.

6.74.4.5 void QwtScaleEngine::setReference (double *r*)

Specify a reference point.

Parameters:

r new reference value

The reference point is needed if options `IncludeRef` or `Symmetric` are active. Its default value is 0.0.

Definition at line 384 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::divideScale()`.

6.74.4.6 double QwtScaleEngine::reference () const**Returns:**

the reference value

See also:

[QwtScaleEngine::setReference\(\)](#), [QwtScaleEngine::setAttribute\(\)](#)

Definition at line 393 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::autoScale()`, `QwtLinearScaleEngine::autoScale()`, and `QwtLog10ScaleEngine::divideScale()`.

6.74.4.7 void QwtScaleEngine::setMargins (double *mlo*, double *mhi*)

Specify margins at the scale's endpoints.

Parameters:

mlo minimum distance between the scale's lower boundary and the smallest enclosed value

mhi minimum distance between the scale's upper boundary and the greatest enclosed value

Margins can be used to leave a minimum amount of space between the enclosed intervals and the boundaries of the scale.

Warning:

- [QwtLog10ScaleEngine](#) measures the margins in decades.

See also:

[QwtScaleEngine::hiMargin](#), [QwtScaleEngine::loMargin](#)

Definition at line 219 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::divideScale()`.

6.74.4.8 double QwtScaleEngine::loMargin () const**Returns:**

the margin at the lower end of the scale The default margin is 0.

See also:

[QwtScaleEngine::setMargins\(\)](#)

Definition at line 187 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::autoScale()`, `QwtLinearScaleEngine::autoScale()`, and `QwtLog10ScaleEngine::divideScale()`.

6.74.4.9 double QwtScaleEngine::hiMargin () const**Returns:**

the margin at the upper end of the scale The default margin is 0.

See also:

[QwtScaleEngine::setMargins\(\)](#)

Definition at line 198 of file `qwt_scale_engine.cpp`.

Referenced by `QwtLog10ScaleEngine::autoScale()`, `QwtLinearScaleEngine::autoScale()`, and `QwtLog10ScaleEngine::divideScale()`.

6.74.4.10 virtual void QwtScaleEngine::autoScale (int *maxNumSteps*, double & *x1*, double & *x2*, double & *stepSize*) const [pure virtual]

Align and divide an interval

Parameters:

maxNumSteps Max. number of steps
x1 First limit of the interval (In/Out)
x2 Second limit of the interval (In/Out)
stepSize Step size (Return value)

Implemented in [QwtLinearScaleEngine](#), and [QwtLog10ScaleEngine](#).

6.74.4.11 virtual [QwtScaleDiv](#) [QwtScaleEngine::divideScale](#) (double *x1*, double *x2*, int *maxMajSteps*, int *maxMinSteps*, double *stepSize* = 0.0) const [pure virtual]

Calculate a scale division.

Parameters:

x1 First interval limit
x2 Second interval limit
maxMajSteps Maximum for the number of major steps
maxMinSteps Maximum number of minor steps
stepSize Step size. If *stepSize* == 0.0, the *scaleEngine* calculates one.

Implemented in [QwtLinearScaleEngine](#), and [QwtLog10ScaleEngine](#).

6.74.4.12 virtual [QwtScaleTransformation*](#) [QwtScaleEngine::transformation](#) () const [pure virtual]

Returns:

a transformation

Implemented in [QwtLinearScaleEngine](#), and [QwtLog10ScaleEngine](#).

6.74.4.13 bool [QwtScaleEngine::contains](#) (const [QwtDoubleInterval](#) & *interval*, double *value*) const [protected]

Check if an interval "contains" a value

Parameters:

interval Interval
value Value

See also:

[QwtScaleArithmetic::compareEps](#)

Definition at line 251 of file `qwt_scale_engine.cpp`.

References [QwtScaleArithmetic::compareEps\(\)](#), [QwtDoubleInterval::isValid\(\)](#), [QwtDoubleInterval::maxValue\(\)](#), [QwtDoubleInterval::minValue\(\)](#), and [QwtDoubleInterval::width\(\)](#).

Referenced by [strip\(\)](#).

6.74.4.14 QwtValueList QwtScaleEngine::strip (const QwtValueList & ticks, const QwtDoubleInterval & interval) const [protected]

Remove ticks from a list, that are not inside an interval

Parameters:

ticks Tick list
interval Interval

Returns:

Stripped tick list

Definition at line 280 of file qwt_scale_engine.cpp.

References contains(), and QwtDoubleInterval::isValid().

6.74.4.15 double QwtScaleEngine::divideInterval (double intervalSize, int numSteps) const [protected]

Calculate a step size for an interval size

Parameters:

intervalSize Interval size
numSteps Number of steps

Returns:

Step size

Definition at line 233 of file qwt_scale_engine.cpp.

References QwtScaleArithmetic::ceil125(), and QwtScaleArithmetic::divideEps().

Referenced by QwtLog10ScaleEngine::autoScale(), QwtLinearScaleEngine::autoScale(), QwtLog10ScaleEngine::divideScale(), and QwtLinearScaleEngine::divideScale().

6.74.4.16 QwtDoubleInterval QwtScaleEngine::buildInterval (double v) const [protected]

Build an interval for a value.

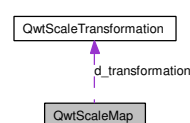
In case of $v == 0.0$ the interval is $[-0.5, 0.5]$, otherwise it is $[0.5 * v, 1.5 * v]$

Definition at line 309 of file qwt_scale_engine.cpp.

Referenced by QwtLog10ScaleEngine::autoScale(), and QwtLinearScaleEngine::autoScale().

6.75 QwtScaleMap Class Reference

Collaboration diagram for QwtScaleMap:



6.75.1 Detailed Description

A scale map.

[QwtScaleMap](#) offers transformations from a scale into a paint interval and vice versa.

Definition at line 55 of file `qwt_scale_map.h`.

Public Member Functions

- [QwtScaleMap](#) ()
- [QwtScaleMap](#) (const [QwtScaleMap](#) &)
- [~QwtScaleMap](#) ()
- [QwtScaleMap](#) & [operator=](#) (const [QwtScaleMap](#) &)
- void [setTransformation](#) ([QwtScaleTransformation](#) *)
- const [QwtScaleTransformation](#) * [transformation](#) () const
- void [setPaintInterval](#) (int p1, int p2)
- void [setPaintXInterval](#) (double p1, double p2)
- void [setScaleInterval](#) (double s1, double s2)
- int [transform](#) (double x) const
- double [invTransform](#) (double i) const
- double [xTransform](#) (double x) const
- double [p1](#) () const
- double [p2](#) () const
- double [s1](#) () const
- double [s2](#) () const
- double [pDist](#) () const
- double [sDist](#) () const

Public Attributes

- QT_STATIC_CONST double [LogMin](#)
- QT_STATIC_CONST double [LogMax](#)

6.75.2 Constructor & Destructor Documentation

6.75.2.1 QwtScaleMap::QwtScaleMap ()

Constructor.

The scale and paint device intervals are both set to [0,1].

Definition at line 84 of file `qwt_scale_map.cpp`.

6.75.2.2 QwtScaleMap::~~QwtScaleMap ()

Destructor

Definition at line 108 of file `qwt_scale_map.cpp`.

6.75.3 Member Function Documentation

6.75.3.1 void QwtScaleMap::setTransformation (QwtScaleTransformation * transformation)

Initialize the map with a transformation

Definition at line 130 of file qwt_scale_map.cpp.

References setScaleInterval(), and transformation().

Referenced by QwtPlot::canvasMap().

6.75.3.2 const QwtScaleTransformation * QwtScaleMap::transformation () const

Get the transformation.

Definition at line 142 of file qwt_scale_map.cpp.

Referenced by QwtPlotScaleItem::draw(), QwtScaleWidget::setScaleDiv(), and setTransformation().

6.75.3.3 void QwtScaleMap::setPaintInterval (int p1, int p2)

Specify the borders of the paint device interval.

Parameters:

p1 first border

p2 second border

Definition at line 180 of file qwt_scale_map.cpp.

References QwtScaleTransformation::type().

Referenced by QwtPlot::canvasMap(), QwtPlotRasterItem::draw(), QwtPainter::drawColorBar(), QwtRoundScaleDraw::QwtRoundScaleDraw(), QwtPlotSpectrogram::renderImage(), QwtRoundScaleDraw::setAngleRange(), and QwtPlotSvgItem::viewBox().

6.75.3.4 void QwtScaleMap::setPaintXInterval (double p1, double p2)

Specify the borders of the paint device interval.

Parameters:

p1 first border

p2 second border

Definition at line 194 of file qwt_scale_map.cpp.

References QwtScaleTransformation::type().

Referenced by QwtPlot::print().

6.75.3.5 void QwtScaleMap::setScaleInterval (double s1, double s2)

Specify the borders of the scale interval.

Parameters:

s1 first border

s2 second border

Warning:

logarithmic scales might be aligned to [LogMin, LogMax]

Definition at line 153 of file `qwt_scale_map.cpp`.

References `LogMax`, `LogMin`, and `QwtScaleTransformation::type()`.

Referenced by `QwtPlot::canvasMap()`, `QwtPlot::print()`, `QwtPlotSpectrogram::renderImage()`, `setTransformation()`, and `QwtPlotSvgItem::viewBox()`.

6.75.3.6 `int QwtScaleMap::transform (double s) const` [inline]

Transform a point related to the scale interval into a point related to the interval of the paint device and round it to an integer. (In Qt <= 3.x paint devices are integer based.)

See also:

[QwtScaleMap::xTransform](#)

Definition at line 175 of file `qwt_scale_map.h`.

References `xTransform()`.

Referenced by `QwtPlotCurve::closePolyline()`, `QwtPlotScaleItem::draw()`, `QwtPlotMarker::draw()`, `QwtPlotSpectrogram::drawContourLines()`, `QwtPlotCurve::drawDots()`, `QwtRoundScaleDraw::drawLabel()`, `QwtPlotCurve::drawLines()`, `QwtPlotCurve::drawSteps()`, `QwtPlotCurve::drawSticks()`, `QwtPlotCurve::drawSymbols()`, `QwtRoundScaleDraw::drawTick()`, `QwtRoundScaleDraw::extent()`, `QwtScaleDraw::labelPosition()`, `QwtPlotPanner::moveCanvas()`, `QwtPlotPicker::transform()`, and `QwtPlotItem::transform()`.

6.75.3.7 `double QwtScaleMap::invTransform (double i) const` [inline]

Transform an paint device value into a value in the interval of the scale.

Definition at line 163 of file `qwt_scale_map.h`.

References `QwtScaleTransformation::invXForm()`.

Referenced by `QwtPainter::drawColorBar()`, `QwtPlotPicker::invTransform()`, `QwtPlotItem::invTransform()`, and `QwtPlotSpectrogram::renderImage()`.

6.75.3.8 `double QwtScaleMap::xTransform (double s) const` [inline]

Transform a point related to the scale interval into a point related to the interval of the paint device

Definition at line 146 of file `qwt_scale_map.h`.

References `QwtScaleTransformation::type()`, and `QwtScaleTransformation::xForm()`.

Referenced by `QwtPlotCurve::closestPoint()`, `QwtPlotCurve::drawLines()`, `transform()`, and `QwtPlotSvgItem::viewBox()`.

6.75.3.9 `double QwtScaleMap::p1 () const` [inline]

Returns:

First border of the paint interval

Definition at line 119 of file qwt_scale_map.h.

Referenced by QwtPlotItem::paintRect(), and QwtPlotSpectrogram::renderImage().

6.75.3.10 double QwtScaleMap::p2 () const [inline]

Returns:

Second border of the paint interval

Definition at line 127 of file qwt_scale_map.h.

Referenced by QwtPlotSpectrogram::renderImage().

6.75.3.11 double QwtScaleMap::s1 () const [inline]

Returns:

First border of the scale interval

Definition at line 103 of file qwt_scale_map.h.

Referenced by QwtPlotSpectrogram::renderImage(), and QwtPlotItem::scaleRect().

6.75.3.12 double QwtScaleMap::s2 () const [inline]

Returns:

Second border of the scale interval

Definition at line 111 of file qwt_scale_map.h.

Referenced by QwtPlotSpectrogram::renderImage().

6.76 QwtScaleTransformation Class Reference

6.76.1 Detailed Description

Operations for linear or logarithmic (base 10) transformations.

Definition at line 19 of file qwt_scale_map.h.

Public Types

- enum [Type](#) {
 RubberBand,
 Text,
 Linear,
 Log10,
 Other }

Public Member Functions

- [QwtScaleTransformation](#) (*Type* type)
- virtual [~QwtScaleTransformation](#) ()
- virtual double [xForm](#) (double x, double s1, double s2, double p1, double p2) const
- virtual double [invXForm](#) (double x, double s1, double s2, double p1, double p2) const
- *Type* type () const
- virtual [QwtScaleTransformation](#) * [copy](#) () const

6.76.2 Constructor & Destructor Documentation**6.76.2.1 QwtScaleTransformation::QwtScaleTransformation (*Type* type)**

Constructor for a linear transformation.

Definition at line 16 of file qwt_scale_map.cpp.

References [QwtScaleTransformation\(\)](#).

Referenced by [QwtScaleTransformation\(\)](#).

6.76.3 Member Function Documentation**6.76.3.1 double QwtScaleTransformation::xForm (double s, double s1, double s2, double p1, double p2) const** [*virtual*]

Transform a value between 2 linear intervals.

Parameters:

- x* value related to the interval [x1, x2]
- x1* first border of source interval
- x2* first border of source interval
- y1* first border of target interval
- y2* first border of target interval

Returns:

- linear mapping:** $y1 + (y2 - y1) / (x2 - x1) * (x - x1)$
- log10 mapping:** $p1 + (p2 - p1) / \log(s2 / s1) * \log(x / s1)$

Definition at line 47 of file qwt_scale_map.cpp.

Referenced by [QwtScaleMap::xTransform\(\)](#).

6.76.3.2 double QwtScaleTransformation::invXForm (double p, double p1, double p2, double s1, double s2) const [*virtual*]

Transform a value from a linear to a logarithmic interval.

Parameters:

- x* value related to the linear interval [p1, p2]
- p1* first border of linear interval

- p2* first border of linear interval
- s1* first border of logarithmic interval
- s2* first border of logarithmic interval

Returns:

$\exp((x - p1) / (p2 - p1) * \log(s2 / s1)) * s1;$

Definition at line 70 of file qwt_scale_map.cpp.

Referenced by QwtScaleMap::invTransform().

6.77 QwtScaleWidget Class Reference

6.77.1 Detailed Description

A Widget which contains a scale.

This Widget can be used to decorate composite widgets with a scale.

Definition at line 34 of file qwt_scale_widget.h.

Signals

- void [scaleDivChanged](#) ()

Public Member Functions

- [QwtScaleWidget](#) (QWidget *parent=NULL)
- [QwtScaleWidget](#) (QwtScaleDraw::Alignment, QWidget *parent=NULL)
- virtual [~QwtScaleWidget](#) ()
- void [setTitle](#) (const QString &title)
- void [setTitle](#) (const [QwtText](#) &title)
- [QwtText](#) [title](#) () const
- void [setBorderDist](#) (int start, int end)
- int [startBorderDist](#) () const
- int [endBorderDist](#) () const
- void [getBorderDistHint](#) (int &start, int &end) const
- void [getMinBorderDist](#) (int &start, int &end) const
- void [setMinBorderDist](#) (int start, int end)
- void [setMargin](#) (int)
- int [margin](#) () const
- void [setSpacing](#) (int td)
- int [spacing](#) () const
- void [setPenWidth](#) (int)
- int [penWidth](#) () const
- void [setScaleDiv](#) (QwtScaleTransformation *, const [QwtScaleDiv](#) &sd)
- void [setScaleDraw](#) (QwtScaleDraw *)
- const [QwtScaleDraw](#) * [scaleDraw](#) () const
- [QwtScaleDraw](#) * [scaleDraw](#) ()
- void [setLabelAlignment](#) (Qt::Alignment)

- void [setLabelRotation](#) (double rotation)
- void [setColorBarEnabled](#) (bool)
- bool [isColorBarEnabled](#) () const
- void [setColorBarWidth](#) (int)
- int [colorBarWidth](#) () const
- void [setColorMap](#) (const [QwtDoubleInterval](#) &, const [QwtColorMap](#) &)
- [QwtDoubleInterval](#) [colorBarInterval](#) () const
- const [QwtColorMap](#) & [colorMap](#) () const
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- int [titleHeightForWidth](#) (int width) const
- int [dimForLength](#) (int length, const QFont &scaleFont) const
- void [drawColorBar](#) (QPainter *painter, const QRect &rect) const
- void [drawTitle](#) (QPainter *painter, [QwtScaleDraw::Alignment](#), const QRect &rect) const
- void [setAlignment](#) ([QwtScaleDraw::Alignment](#))
- [QwtScaleDraw::Alignment](#) [alignment](#) () const
- QRect [colorBarRect](#) (const QRect &) const

Protected Member Functions

- virtual void [paintEvent](#) (QPaintEvent *e)
- virtual void [resizeEvent](#) (QResizeEvent *e)
- void [draw](#) (QPainter *p) const
- void [scaleChange](#) ()
- void [layoutScale](#) (bool update=true)

6.77.2 Constructor & Destructor Documentation

6.77.2.1 QwtScaleWidget::QwtScaleWidget (QWidget *parent = NULL) [explicit]

Create a scale with the position QwtScaleWidget::Left.

Parameters:

parent Parent widget

Definition at line 63 of file qwt_scale_widget.cpp.

6.77.2.2 QwtScaleWidget::QwtScaleWidget ([QwtScaleDraw::Alignment](#) align, QWidget *parent = NULL) [explicit]

Constructor.

Parameters:

align Alignment.

parent Parent widget

Definition at line 87 of file qwt_scale_widget.cpp.

6.77.2.3 QwtScaleWidget::~~QwtScaleWidget () [virtual]

Destructor.

Definition at line 95 of file qwt_scale_widget.cpp.

6.77.3 Member Function Documentation

6.77.3.1 void QwtScaleWidget::scaleDivChanged () [signal]

Signal emitted, whenever the scale division changes.

Referenced by setScaleDiv().

6.77.3.2 void QwtScaleWidget::setTitle (const [QwtText](#) & title)

Give title new text contents.

Parameters:

title New title

See also:

[QwtScaleWidget::title](#)

Warning:

The title flags are interpreted in direction of the label, AlignTop, AlignBottom can't be set as the title will always be aligned to the scale.

Definition at line 167 of file qwt_scale_widget.cpp.

References layoutScale(), [QwtText::renderFlags\(\)](#), [QwtText::setRenderFlags\(\)](#), and [title\(\)](#).

6.77.3.3 [QwtText](#) QwtScaleWidget::title () const

Returns:

title

See also:

[QwtScaleWidget::setTitle](#)

Definition at line 357 of file qwt_scale_widget.cpp.

Referenced by [QwtPlotPrintFilter::apply\(\)](#), [QwtPlot::axisTitle\(\)](#), [drawTitle\(\)](#), and [setTitle\(\)](#).

6.77.3.4 void QwtScaleWidget::setBorderDist (int *dist1*, int *dist2*)

Specify distances of the scale's endpoints from the widget's borders. The actual borders will never be less than minimum border distance.

Parameters:

dist1 Left or top Distance

dist2 Right or bottom distance

See also:

[QwtScaleWidget::borderDist](#)

Definition at line 233 of file `qwt_scale_widget.cpp`.

References [layoutScale\(\)](#).

Referenced by [QwtPlot::updateAxes\(\)](#).

6.77.3.5 `int QwtScaleWidget::startBorderDist () const`

Returns:

start border distance

See also:

[QwtScaleWidget::setBorderDist](#)

Definition at line 366 of file `qwt_scale_widget.cpp`.

Referenced by [QwtPlot::canvasMap\(\)](#), and [QwtPlot::print\(\)](#).

6.77.3.6 `int QwtScaleWidget::endBorderDist () const`

Returns:

end border distance

See also:

[QwtScaleWidget::setBorderDist](#)

Definition at line 375 of file `qwt_scale_widget.cpp`.

Referenced by [QwtPlot::canvasMap\(\)](#), and [QwtPlot::print\(\)](#).

6.77.3.7 `void QwtScaleWidget::getBorderDistHint (int & start, int & end) const`

Calculate a hint for the border distances.

This member function calculates the distance of the scale's endpoints from the widget borders which is required for the mark labels to fit into the widget. The maximum of this distance and the minimum border distance is returned.

Warning:

- The minimum border distance depends on the font.

See also:

[setMinBorderDist\(\)](#), [getMinBorderDist\(\)](#), [setBorderDist\(\)](#)

Definition at line 774 of file `qwt_scale_widget.cpp`.

Referenced by [layoutScale\(\)](#), [minimumSizeHint\(\)](#), and [QwtPlot::updateAxes\(\)](#).

6.77.3.8 void QwtScaleWidget::getMinBorderDist (int & *start*, int & *end*) const

Get the minimum value for the distances of the scale's endpoints from the widget borders.

See also:

[setMinBorderDist\(\)](#), [getBorderDistHint\(\)](#)

Definition at line 805 of file `qwt_scale_widget.cpp`.

6.77.3.9 void QwtScaleWidget::setMinBorderDist (int *start*, int *end*)

Set a minimum value for the distances of the scale's endpoints from the widget borders. This is useful to avoid that the scales are "jumping", when the tick labels or their positions change often.

See also:

[getMinBorderDist\(\)](#), [getBorderDistHint\(\)](#)

Definition at line 793 of file `qwt_scale_widget.cpp`.

6.77.3.10 void QwtScaleWidget::setMargin (int *margin*)

Specify the margin to the colorBar/base line.

Parameters:

margin Margin

See also:

[QwtScaleWidget::margin](#)

Definition at line 248 of file `qwt_scale_widget.cpp`.

References `layoutScale()`.

Referenced by `QwtPlot::print()`.

6.77.3.11 int QwtScaleWidget::margin () const

Returns:

margin

See also:

[QwtScaleWidget::setMargin](#)

Definition at line 384 of file `qwt_scale_widget.cpp`.

Referenced by `QwtPlot::print()`.

6.77.3.12 void QwtScaleWidget::setSpacing (int *spacing*)

Specify the distance between color bar, scale and title.

Parameters:

spacing Spacing

See also:

[QwtScaleWidget::spacing](#)

Definition at line 263 of file qwt_scale_widget.cpp.

References [layoutScale\(\)](#).

6.77.3.13 int QwtScaleWidget::spacing () const**Returns:**

distance between scale and title

See also:

[QwtScaleWidget::setMargin](#)

Definition at line 393 of file qwt_scale_widget.cpp.

Referenced by [QwtPlot::printScale\(\)](#).

6.77.3.14 void QwtScaleWidget::setPenWidth (int *width*)

Specify the width of the scale pen.

Parameters:

width Pen width

See also:

[QwtScaleWidget::penWidth](#)

Definition at line 278 of file qwt_scale_widget.cpp.

References [layoutScale\(\)](#).

6.77.3.15 int QwtScaleWidget::penWidth () const**Returns:**

Scale pen width

See also:

[QwtScaleWidget::setPenWidth](#)

Definition at line 402 of file qwt_scale_widget.cpp.

Referenced by [QwtPlot::printScale\(\)](#).

6.77.3.16 void QwtScaleWidget::setScaleDiv ([QwtScaleTransformation](#) * *transformation*, const [QwtScaleDiv](#) & *scaleDiv*)

Assign a scale division.

The scale division determines where to set the tick marks.

Parameters:

transformation Transformation, needed to translate between scale and pixal values

scaleDiv Scale Division

See also:

For more information about scale divisions, see [QwtScaleDiv](#).

Definition at line 839 of file `qwt_scale_widget.cpp`.

References `layoutScale()`, `QwtAbstractScaleDraw::map()`, `QwtAbstractScaleDraw::scaleDiv()`, `scaleDiv-Changed()`, `QwtAbstractScaleDraw::setScaleDiv()`, `QwtAbstractScaleDraw::setTransformation()`, `QwtScaleMap::transformation()`, and `QwtScaleTransformation::type()`.

Referenced by `QwtPlot::updateAxes()`.

6.77.3.17 void QwtScaleWidget::setScaleDraw ([QwtScaleDraw](#) *)

Set a scale draw `sd` has to be created with new and will be deleted in `QwtScaleWidget::~~QwtScale` or the next call of [QwtScaleWidget::setScaleDraw](#).

Definition at line 321 of file `qwt_scale_widget.cpp`.

References `layoutScale()`, and `QwtScaleDraw::setAlignment()`.

Referenced by `QwtPlot::setAxisScaleDraw()`.

6.77.3.18 const [QwtScaleDraw](#) * QwtScaleWidget::scaleDraw () const

scaleDraw of this scale

See also:

`QwtScaleDraw::setScaleDraw`

Definition at line 339 of file `qwt_scale_widget.cpp`.

Referenced by `alignment()`, `QwtPlot::axisScaleDraw()`, `QwtPlot::printScale()`, and `QwtPlot::sizeHint()`.

6.77.3.19 [QwtScaleDraw](#) * QwtScaleWidget::scaleDraw ()

scaleDraw of this scale

See also:

`QwtScaleDraw::setScaleDraw`

Definition at line 348 of file `qwt_scale_widget.cpp`.

6.77.3.20 void QwtScaleWidget::setLabelAlignment (Qt::Alignment *alignment*)

Change the alignment for the labels.

See also:

[QwtScaleDraw::setLabelAlignment\(\)](#), [QwtScaleWidget::setLabelRotation\(\)](#)

Definition at line 298 of file `qwt_scale_widget.cpp`.

6.77.3.21 void QwtScaleWidget::setLabelRotation (double *rotation*)

Change the rotation for the labels. See [QwtScaleDraw::setLabelRotation\(\)](#).

See also:

[QwtScaleDraw::setLabelRotation\(\)](#), [QwtScaleWidget::setLabelFlags\(\)](#)

Definition at line 310 of file `qwt_scale_widget.cpp`.

References `layoutScale()`.

Referenced by `QwtPlot::setAxisLabelRotation()`.

6.77.3.22 QSize QwtScaleWidget::sizeHint () const [virtual]

Returns:

a size hint

Definition at line 689 of file `qwt_scale_widget.cpp`.

References `minimumSizeHint()`.

6.77.3.23 QSize QwtScaleWidget::minimumSizeHint () const [virtual]

Returns:

a minimum size hint

Definition at line 697 of file `qwt_scale_widget.cpp`.

References `dimForLength()`, and `getBorderDistHint()`.

Referenced by `QwtPlotLayout::minimumSizeHint()`, `sizeHint()`, and `QwtPlot::sizeHint()`.

6.77.3.24 int QwtScaleWidget::titleHeightForWidth (int *width*) const

Find the height of the title for a given width.

Parameters:

width Width

Returns:

height Height

Definition at line 732 of file `qwt_scale_widget.cpp`.

Referenced by `dimForLength()`.

6.77.3.25 `int QwtScaleWidget::dimForLength (int length, const QFont & scaleFont) const`

Find the minimum dimension for a given length. *dim* is the height, *length* the width seen in direction of the title.

Parameters:

length width for horizontal, height for vertical scales
scaleFont Font of the scale

Returns:

height for horizontal, width for vertical scales

Definition at line 746 of file `qwt_scale_widget.cpp`.

References `titleHeightForWidth()`.

Referenced by `minimumSizeHint()`.

6.77.3.26 `void QwtScaleWidget::drawTitle (QPainter * painter, QwtScaleDraw::Alignment align, const QRect & rect) const`

Rotate and paint a title according to its position into a given rectangle.

Parameters:

painter Painter
align Alignment
rect Bounding rectangle

Definition at line 621 of file `qwt_scale_widget.cpp`.

References `QwtText::draw()`, `QwtText::setRenderFlags()`, and `title()`.

Referenced by `draw()`, and `QwtPlot::printScale()`.

6.77.3.27 `void QwtScaleWidget::setAlignment (QwtScaleDraw::Alignment alignment)`

Change the alignment

Parameters:

alignment New alignment

See also:

[QwtScaleWidget::alignment](#)

Definition at line 186 of file `qwt_scale_widget.cpp`.

References `layoutScale()`.

6.77.3.28 `QwtScaleDraw::Alignment QwtScaleWidget::alignment () const`**Returns:**

position

See also:

QwtScaleWidget::setPosition

Definition at line 217 of file qwt_scale_widget.cpp.

References QwtScaleDraw::alignment(), and scaleDraw().

6.77.3.29 void QwtScaleWidget::paintEvent (QPaintEvent * *e*) [protected, virtual]

paintEvent

Definition at line 409 of file qwt_scale_widget.cpp.

References draw().

6.77.3.30 void QwtScaleWidget::resizeEvent (QResizeEvent * *e*) [protected, virtual]

resizeEvent

Definition at line 541 of file qwt_scale_widget.cpp.

References layoutScale().

6.77.3.31 void QwtScaleWidget::draw (QPainter * *p*) const [protected]

draw the scale

Definition at line 427 of file qwt_scale_widget.cpp.

References colorBarRect(), drawColorBar(), and drawTitle().

Referenced by paintEvent().

6.77.3.32 void QwtScaleWidget::scaleChange () [protected]

Notify a change of the scale.

This virtual function can be overloaded by derived classes. The default implementation updates the geometry and repaints the widget.

Definition at line 681 of file qwt_scale_widget.cpp.

References layoutScale().

6.77.3.33 void QwtScaleWidget::layoutScale (bool *update* = true) [protected]

Recalculate the scale's geometry and layout based on.

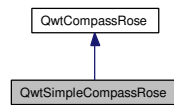
Definition at line 551 of file qwt_scale_widget.cpp.

References colorBarWidth(), and getBorderDistHint().

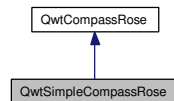
Referenced by resizeEvent(), scaleChange(), setAlignment(), setBorderDist(), setColorBarEnabled(), setColorBarWidth(), setColorMap(), setLabelRotation(), setMargin(), setPenWidth(), setScaleDiv(), setScaleDraw(), setSpacing(), and setTitle().

6.78 QwtSimpleCompassRose Class Reference

Inheritance diagram for QwtSimpleCompassRose:



Collaboration diagram for QwtSimpleCompassRose:



6.78.1 Detailed Description

A simple rose for [QwtCompass](#).

Definition at line 49 of file `qwt_compass_rose.h`.

Public Member Functions

- [QwtSimpleCompassRose](#) (int numThorns=8, int numThornLevels=-1)
- void [setWidth](#) (double w)
- double [width](#) () const
- void [setNumThorns](#) (int count)
- int [numThorns](#) () const
- void [setNumThornLevels](#) (int count)
- int [numThornLevels](#) () const
- void [setShrinkFactor](#) (double factor)
- double [shrinkFactor](#) () const
- virtual void [draw](#) (QPainter *, const QPoint ¢er, int radius, double north, QPalette::ColorGroup=QPalette::Active) const

Static Public Member Functions

- static void [drawRose](#) (QPainter *, const QPalette &, const QPoint ¢er, int radius, double origin, double width, int numThorns, int numThornLevels, double shrinkFactor)

6.78.2 Constructor & Destructor Documentation

6.78.2.1 QwtSimpleCompassRose::QwtSimpleCompassRose (int *numThorns* = 8, int *numThornLevels* = -1)

Constructor

Parameters:

numThorns Number of thorns

numThornLevels Number of thorn levels

Definition at line 61 of file `qwt_compass_rose.cpp`.

References [QwtCompassRose::palette\(\)](#), and [QwtCompassRose::setPalette\(\)](#).

6.78.3 Member Function Documentation

6.78.3.1 void QwtSimpleCompassRose::setWidth (double *width*)

Set the width of the rose heads. Lower value make thinner heads. The range is limited from 0.03 to 0.4.

Parameters:

width Width

Definition at line 226 of file qwt_compass_rose.cpp.

6.78.3.2 void QwtSimpleCompassRose::setNumThorns (int *numThorns*)

Set the number of thorns on one level The number is aligned to a multiple of 4, with a minimum of 4

Parameters:

numThorns Number of thorns

See also:

[numThorns\(\)](#), [setNumThornLevels\(\)](#)

Definition at line 243 of file qwt_compass_rose.cpp.

6.78.3.3 int QwtSimpleCompassRose::numThorns () const

Returns:

Number of thorns

See also:

[setNumThorns\(\)](#), [setNumThornLevels\(\)](#)

Definition at line 258 of file qwt_compass_rose.cpp.

6.78.3.4 void QwtSimpleCompassRose::setNumThornLevels (int *numThornLevels*)

Set the of thorns levels

Parameters:

numThornLevels Number of thorns levels

See also:

[setNumThorns\(\)](#), [numThornLevels\(\)](#)

Definition at line 269 of file qwt_compass_rose.cpp.

6.78.3.5 int QwtSimpleCompassRose::numThornLevels () const

Returns:

Number of thorn levels

See also:

[setNumThorns\(\)](#), [setNumThornLevels\(\)](#)

Definition at line 278 of file qwt_compass_rose.cpp.

6.78.3.6 void QwtSimpleCompassRose::draw (QPainter * *painter*, const QPoint & *center*, int *radius*, double *north*, QPalette::ColorGroup *cg* = QPalette::Active) const [virtual]

Draw the rose

Parameters:

painter Painter
center Center point
radius Radius of the rose
north Position
cg Color group

Implements [QwtCompassRose](#).

Definition at line 98 of file qwt_compass_rose.cpp.

References [drawRose\(\)](#), and [QwtCompassRose::palette\(\)](#).

6.78.3.7 void QwtSimpleCompassRose::drawRose (QPainter * *painter*, const QPalette & *palette*, const QPoint & *center*, int *radius*, double *north*, double *width*, int *numThorns*, int *numThornLevels*, double *shrinkFactor*) [static]

Draw the rose

Parameters:

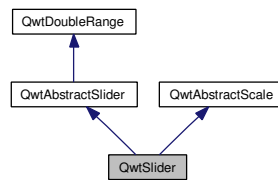
painter Painter
palette Palette
center Center of the rose
radius Radius of the rose
north Position pointing to north
width Width of the rose
numThorns Number of thorns
numThornLevels Number of thorn levels
shrinkFactor Factor to shrink the thorns with each level

Definition at line 136 of file qwt_compass_rose.cpp.

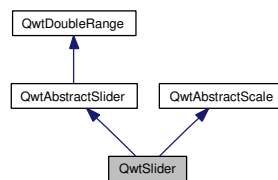
Referenced by [draw\(\)](#).

6.79 QwtSlider Class Reference

Inheritance diagram for QwtSlider:



Collaboration diagram for QwtSlider:



6.79.1 Detailed Description

The Slider Widget.

[QwtSlider](#) is a slider widget which operates on an interval of type double. [QwtSlider](#) supports different layouts as well as a scale.

See also:

[QwtAbstractSlider](#) and [QwtAbstractScale](#) for the descriptions of the inherited members.

Definition at line 34 of file qwt_slider.h.

Public Types

- enum [ScalePos](#) {
 - NoScale,**
 - LeftScale,**
 - RightScale,**
 - TopScale,**
 - BottomScale,**
 - NoScale,**
 - LeftScale,**
 - RightScale,**
 - TopScale,**
 - BottomScale }**

- enum [BGSTYLE](#) {
BgTrough = 0x1,
BgSlot = 0x2,
BgBoth = BgTrough | BgSlot }

Public Member Functions

- [QwtSlider](#) (QWidget *parent, Qt::Orientation=Qt::Horizontal, [ScalePos](#)=NoScale, [BGSTYLE](#) bgStyle=BgTrough)
- virtual [~QwtSlider](#) ()
- virtual void [setOrientation](#) (Qt::Orientation)
- void [setBgStyle](#) ([BGSTYLE](#))
- [BGSTYLE](#) [bgStyle](#) () const
- void [setScalePosition](#) ([ScalePos](#) s)
- [ScalePos](#) [scalePosition](#) () const
- int [thumbLength](#) () const
- int [thumbWidth](#) () const
- int [borderWidth](#) () const
- void [setThumbLength](#) (int l)
- void [setThumbWidth](#) (int w)
- void [setBorderWidth](#) (int bw)
- void [setMargins](#) (int x, int y)
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- void [setScaleDraw](#) ([QwtScaleDraw](#) *)
- const [QwtScaleDraw](#) * [scaleDraw](#) () const

Protected Member Functions

- virtual double [getValue](#) (const QPoint &p)
- virtual void [getScrollMode](#) (const QPoint &p, int &scrollMode, int &direction)
- void [draw](#) (QPainter *p, const QRect &update_rect)
- virtual void [drawSlider](#) (QPainter *p, const QRect &r)
- virtual void [drawThumb](#) (QPainter *p, const QRect &, int pos)
- virtual void [resizeEvent](#) (QResizeEvent *e)
- virtual void [paintEvent](#) (QPaintEvent *e)
- virtual void [valueChange](#) ()
- virtual void [rangeChange](#) ()
- virtual void [scaleChange](#) ()
- virtual void [fontChange](#) (const QFont &oldFont)
- void [layoutSlider](#) (bool update=true)
- int [xyPosition](#) (double v) const
- [QwtScaleDraw](#) * [scaleDraw](#) ()

6.79.2 Member Enumeration Documentation

6.79.2.1 enum [QwtSlider::ScalePos](#)

Scale position. [QwtSlider](#) tries to enforce valid combinations of its orientation and scale position:

- Qt::Horizontal combines with NoScale, TopScale and BottomScale
- Qt::Vertical combines with NoScale, LeftScale and RightScale

See also:

[QwtSlider::QwtSlider](#)

Definition at line 56 of file `qwt_slider.h`.

6.79.2.2 enum [QwtSlider::BGSTYLE](#)

Background style.

See also:

[QwtSlider::QwtSlider](#)

Definition at line 70 of file `qwt_slider.h`.

6.79.3 Constructor & Destructor Documentation

6.79.3.1 [QwtSlider::QwtSlider](#) ([QWidget](#) * *parent*, [Qt::Orientation](#) *orientation* = [Qt::Horizontal](#), [ScalePos](#) *scalePos* = NoScale, [BGSTYLE](#) *bgStyle* = BgTrough) [explicit]

Constructor.

Parameters:

parent parent widget

orientation Orientation of the slider. Can be Qt::Horizontal or Qt::Vertical. Defaults to Qt::Horizontal.

scalePos Position of the scale. Defaults to [QwtSlider::NoScale](#).

bgStyle Background style. [QwtSlider::BgTrough](#) draws the slider button in a trough, [QwtSlider::BgSlot](#) draws a slot underneath the button. An or-combination of both may also be used. The default is [QwtSlider::BgTrough](#).

[QwtSlider](#) enforces valid combinations of its orientation and scale position. If the combination is invalid, the scale position will be set to NoScale. Valid combinations are:

- Qt::Horizontal with NoScale, TopScale, or BottomScale;
- Qt::Vertical with NoScale, LeftScale, or RightScale.

Definition at line 64 of file `qwt_slider.cpp`.

6.79.4 Member Function Documentation

6.79.4.1 void QwtSlider::setOrientation (Qt::Orientation *o*) [virtual]

Set the orientation.

Parameters:

o Orientation. Allowed values are Qt::Horizontal and Qt::Vertical.

If the new orientation and the old scale position are an invalid combination, the scale position will be set to QwtSlider::NoScale.

See also:

[QwtAbstractSlider::orientation\(\)](#)

Reimplemented from [QwtAbstractSlider](#).

Definition at line 174 of file qwt_slider.cpp.

References [layoutSlider\(\)](#), [QwtAbstractSlider::orientation\(\)](#), and [QwtAbstractSlider::setOrientation\(\)](#).

Referenced by [setScalePosition\(\)](#).

6.79.4.2 void QwtSlider::setBgStyle (BGSTYLE *st*)

Set the background style.

Definition at line 784 of file qwt_slider.cpp.

References [layoutSlider\(\)](#).

6.79.4.3 QwtSlider::BGSTYLE QwtSlider::bgStyle () const

Returns:

the background style.

Definition at line 793 of file qwt_slider.cpp.

6.79.4.4 void QwtSlider::setScalePosition (ScalePos *s*)

Change the scale position (and slider orientation).

Parameters:

s Position of the scale.

A valid combination of scale position and orientation is enforced:

- if the new scale position is Left or Right, the scale orientation will become Qt::Vertical;
- if the new scale position is Bottom or Top the scale orientation will become Qt::Horizontal;
- if the new scale position is QwtSlider::NoScale, the scale orientation will not change.

Definition at line 224 of file qwt_slider.cpp.

References [layoutSlider\(\)](#), [scaleDraw\(\)](#), [QwtScaleDraw::setAlignment\(\)](#), and [setOrientation\(\)](#).

6.79.4.5 QwtSlider::ScalePos QwtSlider::scalePosition () const

Return the scale position.

Definition at line 267 of file qwt_slider.cpp.

6.79.4.6 int QwtSlider::thumbLength () const**Returns:**

the thumb length.

Definition at line 801 of file qwt_slider.cpp.

Referenced by getScrollMode().

6.79.4.7 int QwtSlider::thumbWidth () const**Returns:**

the thumb width.

Definition at line 809 of file qwt_slider.cpp.

6.79.4.8 int QwtSlider::borderWidth () const**Returns:**

the border width.

Definition at line 817 of file qwt_slider.cpp.

6.79.4.9 void QwtSlider::setThumbLength (int *thumbLength*)

Set the slider's thumb length.

Parameters:

thumbLength new length

Definition at line 292 of file qwt_slider.cpp.

References layoutSlider().

6.79.4.10 void QwtSlider::setThumbWidth (int *w*)

Change the width of the thumb.

Parameters:

w new width

Definition at line 308 of file qwt_slider.cpp.

References layoutSlider().

6.79.4.11 void QwtSlider::setBorderWidth (int *bd*)

Change the slider's border width.

Parameters:

bd border width

Definition at line 276 of file qwt_slider.cpp.

References layoutSlider().

6.79.4.12 void QwtSlider::setMargins (int *xMargin*, int *yMargin*)

Set distances between the widget's border and internals.

Parameters:

xMargin Horizontal margin

yMargin Vertical margin

Definition at line 766 of file qwt_slider.cpp.

References layoutSlider().

6.79.4.13 QSize QwtSlider::sizeHint () const [virtual]**Returns:**

[QwtSlider::minimumSizeHint\(\)](#)

Definition at line 825 of file qwt_slider.cpp.

References minimumSizeHint().

6.79.4.14 QSize QwtSlider::minimumSizeHint () const [virtual]

Return a minimum size hint.

Warning:

The return value of [QwtSlider::minimumSizeHint\(\)](#) depends on the font and the scale.

Definition at line 835 of file qwt_slider.cpp.

References [QwtScaleDraw::extent\(\)](#), [QwtScaleDraw::getBorderDistHint\(\)](#), [QwtScaleDraw::minLength\(\)](#), [QwtAbstractSlider::orientation\(\)](#), and [scaleDraw\(\)](#).

Referenced by [sizeHint\(\)](#).

6.79.4.15 void QwtSlider::setScaleDraw ([QwtScaleDraw](#) * *scaleDraw*)

Set a scale draw.

For changing the labels of the scales, it is necessary to derive from [QwtScaleDraw](#) and overload [QwtScaleDraw::label\(\)](#).

Parameters:

scaleDraw ScaleDraw object, that has to be created with new and will be deleted in ~QwtSlider or the next call of [setScaleDraw\(\)](#).

Definition at line 331 of file qwt_slider.cpp.

References [QwtScaleDraw::alignment\(\)](#), [layoutSlider\(\)](#), [scaleDraw\(\)](#), [QwtAbstractScale::setAbstractScaleDraw\(\)](#), and [QwtScaleDraw::setAlignment\(\)](#).

6.79.4.16 const [QwtScaleDraw](#) * QwtSlider::scaleDraw () const**Returns:**

the scale draw of the slider

See also:

[setScaleDraw\(\)](#)

Definition at line 348 of file qwt_slider.cpp.

References [QwtAbstractScale::abstractScaleDraw\(\)](#).

Referenced by [draw\(\)](#), [getScrollMode\(\)](#), [layoutSlider\(\)](#), [minimumSizeHint\(\)](#), [setScaleDraw\(\)](#), and [setScalePosition\(\)](#).

6.79.4.17 double QwtSlider::getValue (const QPoint & *p*) [protected, virtual]

Determine the value corresponding to a specified mouse location.

Implements [QwtAbstractSlider](#).

Definition at line 517 of file qwt_slider.cpp.

References [QwtAbstractSlider::orientation\(\)](#).

6.79.4.18 void QwtSlider::getScrollMode (const QPoint & *p*, int & *scrollMode*, int & *direction*) [protected, virtual]

Determine scrolling mode and direction.

Parameters:

p point

scrollMode Scrolling mode

direction Direction

Implements [QwtAbstractSlider](#).

Definition at line 530 of file qwt_slider.cpp.

References [QwtAbstractSlider::orientation\(\)](#), [scaleDraw\(\)](#), [thumbLength\(\)](#), [QwtDoubleRange::value\(\)](#), and [xyPosition\(\)](#).

6.79.4.19 void QwtSlider::draw (QPainter **p*, const QRect & *update_rect*) [protected]

Draw the [QwtSlider](#).

Definition at line 575 of file qwt_slider.cpp.

References [QwtAbstractScaleDraw::draw\(\)](#), [QwtPainter::drawFocusRect\(\)](#), [drawSlider\(\)](#), and [scaleDraw\(\)](#).

Referenced by [paintEvent\(\)](#).

6.79.4.20 void QwtSlider::drawSlider (QPainter * *p*, const QRect & *r*) [protected, virtual]

Draw the slider into the specified rectangle.

Definition at line 377 of file qwt_slider.cpp.

References [drawThumb\(\)](#), [QwtAbstractSlider::isValid\(\)](#), [QwtAbstractSlider::orientation\(\)](#), [QwtDoubleRange::value\(\)](#), and [xyPosition\(\)](#).

Referenced by [draw\(\)](#).

6.79.4.21 void QwtSlider::drawThumb (QPainter * *p*, const QRect &, int *pos*) [protected, virtual]

Draw the thumb at a position.

Definition at line 453 of file qwt_slider.cpp.

References [QwtAbstractSlider::orientation\(\)](#).

Referenced by [drawSlider\(\)](#).

6.79.4.22 void QwtSlider::resizeEvent (QResizeEvent * *e*) [protected, virtual]

Qt resize event.

Definition at line 593 of file qwt_slider.cpp.

References [layoutSlider\(\)](#).

6.79.4.23 void QwtSlider::paintEvent (QPaintEvent * *e*) [protected, virtual]

Qt paint event.

Definition at line 559 of file qwt_slider.cpp.

References [draw\(\)](#).

6.79.4.24 void QwtSlider::valueChange () [protected, virtual]

Notify change of value.

Reimplemented from [QwtAbstractSlider](#).

Definition at line 742 of file qwt_slider.cpp.

References [QwtAbstractSlider::valueChange\(\)](#).

6.79.4.25 void QwtSlider::rangeChange () [protected, virtual]

Notify change of range.

Reimplemented from [QwtDoubleRange](#).

Definition at line 750 of file qwt_slider.cpp.

References [QwtAbstractScale::autoScale\(\)](#), [layoutSlider\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), [QwtDoubleRange::rangeChange\(\)](#), and [QwtAbstractScale::rescale\(\)](#).

6.79.4.26 void QwtSlider::scaleChange () [protected, virtual]

Notify changed scale.

Reimplemented from [QwtAbstractScale](#).

Definition at line 363 of file qwt_slider.cpp.

References [layoutSlider\(\)](#).

6.79.4.27 void QwtSlider::fontChange (const QFont & oldFont) [protected, virtual]

Notify change in font.

Definition at line 370 of file qwt_slider.cpp.

References [layoutSlider\(\)](#).

6.79.4.28 void QwtSlider::layoutSlider (bool update_geometry = true) [protected]

Recalculate the slider's geometry and layout based on the current rect and fonts.

Parameters:

update_geometry notify the layout system and call update to redraw the scale

Definition at line 604 of file qwt_slider.cpp.

References [QwtScaleDraw::getBorderDistHint\(\)](#), [QwtScaleDraw::move\(\)](#), [QwtAbstractSlider::orientation\(\)](#), [scaleDraw\(\)](#), and [QwtScaleDraw::setLength\(\)](#).

Referenced by [fontChange\(\)](#), [rangeChange\(\)](#), [resizeEvent\(\)](#), [scaleChange\(\)](#), [setBgStyle\(\)](#), [setBorderWidth\(\)](#), [setMargins\(\)](#), [setOrientation\(\)](#), [setScaleDraw\(\)](#), [setScalePosition\(\)](#), [setThumbLength\(\)](#), and [setThumbWidth\(\)](#).

6.79.4.29 int QwtSlider::xyPosition (double v) const [protected]

Find the x/y position for a given value v.

Definition at line 511 of file qwt_slider.cpp.

Referenced by [drawSlider\(\)](#), and [getScrollMode\(\)](#).

6.79.4.30 QwtScaleDraw * QwtSlider::scaleDraw () [protected]

Returns:

the scale draw of the slider

See also:

[setScaleDraw\(\)](#)

Definition at line 357 of file qwt_slider.cpp.

References `QwtAbstractScale::abstractScaleDraw()`.

6.80 QwtSpline Class Reference

6.80.1 Detailed Description

A class for spline interpolation.

The [QwtSpline](#) class is used for cubical spline interpolation. Two types of splines, natural and periodic, are supported.

Usage:

1. First call [setPoints\(\)](#) to determine the spline coefficients for a tabulated function $y(x)$.
2. After the coefficients have been set up, the interpolated function value for an argument x can be determined by calling [QwtSpline::value\(\)](#).

Example:

```
#include <qwt_spline.h>

QPolygonF interpolate(const QPolygonF& points, int numValues)
{
    QwtSpline spline;
    if ( !spline.setPoints(points) )
        return points;

    QPolygonF interpolatedPoints(numValues);

    const double delta =
        (points[numPoints - 1].x() - points[0].x()) / (points.size() - 1);
    for(i = 0; i < points.size(); i++) / interpolate
    {
        const double x = points[0].x() + i * delta;
        interpolatedPoints[i].setX(x);
        interpolatedPoints[i].setY(spline.value(x));
    }
    return interpolatedPoints;
}
```

Definition at line 77 of file qwt_spline.h.

Public Types

- enum [SplineType](#) {
 Natural,
 Periodic }

Public Member Functions

- [QwtSpline](#) ()
- [QwtSpline](#) (const [QwtSpline](#) &)
- [~QwtSpline](#) ()
- [QwtSpline](#) & [operator=](#) (const [QwtSpline](#) &)

- void [setSplineType](#) ([SplineType](#))
- [SplineType](#) [splineType](#) () const
- bool [setPoints](#) (const QPolygonF &points)
- QPolygonF [points](#) () const
- void [reset](#) ()
- bool [isValid](#) () const
- double [value](#) (double x) const

Protected Member Functions

- bool [buildNaturalSpline](#) (const QPolygonF &)
- bool [buildPeriodicSpline](#) (const QPolygonF &)

Protected Attributes

- PrivateData * [d_data](#)

6.80.2 Constructor & Destructor Documentation

6.80.2.1 QwtSpline::QwtSpline ()

Constructor.

Definition at line 73 of file qwt_spline.cpp.

References [d_data](#).

6.80.2.2 QwtSpline::~QwtSpline ()

Destructor.

Definition at line 90 of file qwt_spline.cpp.

References [d_data](#).

6.80.3 Member Function Documentation

6.80.3.1 bool QwtSpline::setPoints (const QPolygonF & *points*)

Determine the function table index corresponding to a value *x* Calculate the spline coefficients.

Depending on the value of *periodic*, this function will determine the coefficients for a natural or a periodic spline and store them internally.

Parameters:

- x*
- y* points
- size* number of points
- periodic* if true, calculate periodic spline

Returns:

- true if successful

Warning:

The sequence of x (but not y) values has to be strictly monotone increasing, which means $x[0] < x[1] < \dots < x[n-1]$. If this is not the case, the function will return false

Definition at line 126 of file `qwt_spline.cpp`.

6.80.3.2 QPolygonF QwtSpline::points () const

Return points passed by `setPoints`

Definition at line 164 of file `qwt_spline.cpp`.

6.80.3.3 void QwtSpline::reset ()

Free allocated memory and set size to 0.

Definition at line 172 of file `qwt_spline.cpp`.

References `d_data`.

6.80.3.4 bool QwtSpline::isValid () const

True if valid.

Definition at line 181 of file `qwt_spline.cpp`.

References `d_data`.

6.80.3.5 double QwtSpline::value (double x) const

Calculate the interpolated function value corresponding to a given argument x .

Definition at line 190 of file `qwt_spline.cpp`.

References `d_data`.

6.80.3.6 bool QwtSpline::buildNaturalSpline (const QPolygonF & *points*) [protected]

Determines the coefficients for a natural spline.

Returns:

true if successful

Definition at line 209 of file `qwt_spline.cpp`.

6.80.3.7 bool QwtSpline::buildPeriodicSpline (const QPolygonF & *points*) [protected]

Determines the coefficients for a periodic spline.

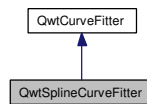
Returns:

true if successful

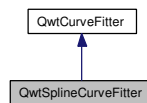
Definition at line 292 of file `qwt_spline.cpp`.

6.81 QwtSplineCurveFitter Class Reference

Inheritance diagram for QwtSplineCurveFitter:



Collaboration diagram for QwtSplineCurveFitter:



6.81.1 Detailed Description

A curve fitter using cubic splines.

Definition at line 65 of file qwt_curve_fitter.h.

Public Types

- enum [FitMode](#) {
 Auto,
 Spline,
 ParametricSpline }

Public Member Functions

- [QwtSplineCurveFitter](#) ()
- virtual [~QwtSplineCurveFitter](#) ()
- void [setFitMode](#) ([FitMode](#))
- [FitMode](#) [fitMode](#) () const
- void [setSpline](#) (const [QwtSpline](#) &)
- const [QwtSpline](#) & [spline](#) () const
- [QwtSpline](#) & [spline](#) ()
- void [setSplineSize](#) (int size)
- int [splineSize](#) () const
- virtual [QPolygonF](#) [fitCurve](#) (const [QPolygonF](#) &) const

6.82 QwtSymbol Class Reference

6.82.1 Detailed Description

A class for drawing symbols.

Definition at line 22 of file qwt_symbol.h.

Public Types

- enum [Style](#) {
 Arrow,
 Ray,
 TriangleStyle,
 ThinStyle,
 Style1,
 Style2,
 NoSymbol = -1,
 Ellipse,
 Rect,
 Diamond,
 Triangle,
 DTriangle,
 UTriangle,
 LTriangle,
 RTriangle,
 Cross,
 XCross,
 HLine,
 VLine,
 Star1,
 Star2,
 Hexagon,
 StyleCnt }

Public Member Functions

- [QwtSymbol](#) ()
- [QwtSymbol](#) ([Style](#) st, const [QBrush](#) &bd, const [QPen](#) &pn, const [QSize](#) &s)
- virtual [~QwtSymbol](#) ()
- bool [operator!=](#) (const [QwtSymbol](#) &) const
- virtual bool [operator==](#) (const [QwtSymbol](#) &) const
- virtual [QwtSymbol](#) * [clone](#) () const
- void [setSize](#) (const [QSize](#) &s)
- void [setSize](#) (int a, int b=-1)
- void [setBrush](#) (const [QBrush](#) &b)
- void [setPen](#) (const [QPen](#) &p)
- void [setStyle](#) ([Style](#) s)
- const [QBrush](#) & [brush](#) () const
- const [QPen](#) & [pen](#) () const
- const [QSize](#) & [size](#) () const
- [Style](#) [style](#) () const
- void [draw](#) ([QPainter](#) *p, const [QPoint](#) &pt) const
- void [draw](#) ([QPainter](#) *p, int x, int y) const
- virtual void [draw](#) ([QPainter](#) *p, const [QRect](#) &r) const

6.82.2 Member Enumeration Documentation

6.82.2.1 enum [QwtSymbol::Style](#)

Style

See also:

[setStyle\(\)](#), [style\(\)](#)

Definition at line 29 of file qwt_symbol.h.

6.82.3 Constructor & Destructor Documentation

6.82.3.1 [QwtSymbol::QwtSymbol \(\)](#)

Default Constructor

The symbol is constructed with gray interior, black outline with zero width, no size and style 'NoSymbol'.

Definition at line 22 of file qwt_symbol.cpp.

6.82.3.2 [QwtSymbol::QwtSymbol \(\[QwtSymbol::Style\]\(#\) style, const QBrush & brush, const QPen & pen, const QSize & size\)](#)

Constructor.

Parameters:

style Symbol Style

brush brush to fill the interior

pen outline pen

size size

Definition at line 37 of file qwt_symbol.cpp.

6.82.3.3 [QwtSymbol::~~QwtSymbol \(\)](#) [virtual]

Destructor.

Definition at line 47 of file qwt_symbol.cpp.

6.82.4 Member Function Documentation

6.82.4.1 [bool QwtSymbol::operator!= \(const \[QwtSymbol\]\(#\) &\) const](#)

!= operator

Definition at line 350 of file qwt_symbol.cpp.

6.82.4.2 [bool QwtSymbol::operator== \(const \[QwtSymbol\]\(#\) &\) const](#) [virtual]

== operator

Definition at line 343 of file qwt_symbol.cpp.

References [brush\(\)](#), [pen\(\)](#), [size\(\)](#), and [style\(\)](#).

6.82.4.3 void QwtSymbol::setSize (const QSize & *s*)

Set the symbol's size.

Definition at line 76 of file qwt_symbol.cpp.

6.82.4.4 void QwtSymbol::setSize (int *w*, int *h* = -1)

Specify the symbol's size.

If the '*h*' parameter is left out or less than 0, and the '*w*' parameter is greater than or equal to 0, the symbol size will be set to (*w*,*w*).

Parameters:

w width

h height (defaults to -1)

Definition at line 68 of file qwt_symbol.cpp.

6.82.4.5 void QwtSymbol::setBrush (const QBrush & *br*)

Assign a brush.

The brush is used to draw the interior of the symbol.

Parameters:

br brush

Definition at line 88 of file qwt_symbol.cpp.

Referenced by QwtPlotPrintFilter::apply(), and QwtPlotPrintFilter::reset().

6.82.4.6 void QwtSymbol::setPen (const QPen & *pn*)

Assign a pen.

The pen is used to draw the symbol's outline.

Parameters:

pn pen

Definition at line 100 of file qwt_symbol.cpp.

Referenced by QwtPlotPrintFilter::apply(), and QwtPlotPrintFilter::reset().

6.82.4.7 void QwtSymbol::setStyle (QwtSymbol::Style *s*)

Specify the symbol style.

The following styles are defined:

NoSymbol No Style. The symbol cannot be drawn.

Ellipse Ellipse or circle

Rect Rectangle

Diamond Diamond

Triangle Triangle pointing upwards

DTriangle Triangle pointing downwards

UTriangle Triangle pointing upwards

LTriangle Triangle pointing left

RTriangle Triangle pointing right

Cross Cross (+)

XCross Diagonal cross (X)

HLine Horizontal line

VLine Vertical line

Star1 X combined with +

Star2 Six-pointed star

Hexagon Hexagon

Parameters:

s style

Definition at line 337 of file qwt_symbol.cpp.

6.82.4.8 const QBrush& QwtSymbol::brush () const [inline]

Return Brush.

Definition at line 69 of file qwt_symbol.h.

Referenced by QwtPlotPrintFilter::apply(), QwtPlotCurve::drawSymbols(), operator==(), and QwtPlotPrintFilter::reset().

6.82.4.9 const QPen& QwtSymbol::pen () const [inline]

Return Pen.

Definition at line 71 of file qwt_symbol.h.

Referenced by QwtPlotPrintFilter::apply(), QwtPlotCurve::drawSymbols(), operator==(), and QwtPlotPrintFilter::reset().

6.82.4.10 const QSize& QwtSymbol::size () const [inline]

Return Size.

Definition at line 73 of file qwt_symbol.h.

Referenced by QwtPlotCurve::drawSymbols(), and operator==().

6.82.4.11 [Style](#) `QwtSymbol::style () const` `[inline]`

Return Style.

Definition at line 75 of file `qwt_symbol.h`.

Referenced by `operator==()`.

6.82.4.12 `void QwtSymbol::draw (QPainter * painter, const QPoint & pos) const`

Draw the symbol at a specified point.

Parameters:

painter Painter

pos Center of the symbol

Definition at line 302 of file `qwt_symbol.cpp`.

References `QwtPainter::metricsMap()`.

Referenced by `draw()`, and `QwtPlotCurve::drawSymbols()`.

6.82.4.13 `void QwtSymbol::draw (QPainter * p, int x, int y) const`

Draw the symbol at a point (*x*,*y*).

Definition at line 108 of file `qwt_symbol.cpp`.

References `draw()`.

6.82.4.14 `void QwtSymbol::draw (QPainter * painter, const QRect & r) const` `[virtual]`

Draw the symbol into a bounding rectangle.

This function assumes that the painter has been initialized with brush and pen before. This allows a much more performant implementation when painting many symbols with the same brush and pen like in curves.

Parameters:

painter Painter

r Bounding rectangle

Definition at line 124 of file `qwt_symbol.cpp`.

References `QwtPainter::drawEllipse()`, `QwtPainter::drawLine()`, `QwtPainter::drawPolygon()`, and `QwtPainter::drawRect()`.

6.83 QwtText Class Reference

6.83.1 Detailed Description

A class representing a text.

A [QwtText](#) is a text including a set of attributes how to render it.

- **Format**

A text might include control sequences (f.e tags) describing how to render it. Each format (f.e MathML, TeX, Qt Rich Text) has its own set of control sequences, that can be handles by a [QwtTextEngine](#) for this format.

- **Background**

A text might have a background, defined by a QPen and QBrush to improve its visibility.

- **Font**

A text might have an individual font.

- **Color**

A text might have an individual color.

- **Render Flags**

Flags from Qt::AlignmentFlag and Qt::TextFlag used like in QPainter::drawText.

See also:

[QwtTextEngine](#), [QwtTextLabel](#)

Definition at line 51 of file qwt_text.h.

Public Types

- enum [TextFormat](#) {
 AutoText = 0,
 PlainText,
 RichText,
 MathMLText,
 TeXText,
 OtherFormat = 100 }
- enum [PaintAttribute](#) {
 PaintCached = 1,
 PaintPacked = 2,
 PaintFiltered = 1,
 ClipPolygons = 2,
 PaintUsingTextFont = 1,
 PaintUsingTextColor = 2,
 PaintBackground = 4 }
- enum [LayoutAttribute](#) { **MinimumLayout** = 1 }

Public Member Functions

- [QwtText](#) (const QString &=QString::null, [TextFormat](#) textFormat=AutoText)
- [QwtText](#) (const [QwtText](#) &)
- [~QwtText](#) ()
- [QwtText](#) & [operator=](#) (const [QwtText](#) &)
- int [operator==](#) (const [QwtText](#) &) const
- int [operator!=](#) (const [QwtText](#) &) const
- void [setText](#) (const QString &, [QwtText::TextFormat](#) textFormat=AutoText)
- QString [text](#) () const
- bool [isNull](#) () const
- bool [isEmpty](#) () const
- void [setFont](#) (const QFont &)
- QFont [font](#) () const
- QFont [usedFont](#) (const QFont &) const
- void [setRenderFlags](#) (int flags)
- int [renderFlags](#) () const
- void [setColor](#) (const QColor &)
- QColor [color](#) () const
- QColor [usedColor](#) (const QColor &) const
- void [setBackgroundPen](#) (const QPen &)
- QPen [backgroundPen](#) () const
- void [setBackgroundBrush](#) (const QBrush &)
- QBrush [backgroundBrush](#) () const
- void [setPaintAttribute](#) ([PaintAttribute](#), bool on=true)
- bool [testPaintAttribute](#) ([PaintAttribute](#)) const
- void [setLayoutAttribute](#) ([LayoutAttribute](#), bool on=true)
- bool [testLayoutAttribute](#) ([LayoutAttribute](#)) const
- int [heightForWidth](#) (int width, const QFont &=QFont()) const
- QSize [textSize](#) (const QFont &=QFont()) const
- void [draw](#) (QPainter *painter, const QRect &rect) const

Static Public Member Functions

- static const [QwtTextEngine](#) * [textEngine](#) (const QString &text, [QwtText::TextFormat](#)=AutoText)
- static const [QwtTextEngine](#) * [textEngine](#) ([QwtText::TextFormat](#))
- static void [setTextEngine](#) ([QwtText::TextFormat](#), [QwtTextEngine](#) *)

6.83.2 Member Enumeration Documentation**6.83.2.1 enum [QwtText::TextFormat](#)**

Text format.

The text format defines the [QwtTextEngine](#), that is used to render the text.

- AutoText

The text format is determined using [QwtTextEngine::mightRender](#) for all available text engines in increasing order > PlainText. If none of the text engines can render the text is rendered like PlainText.

- PlainText
Draw the text as it is, using a [QwtPlainTextEngine](#).
- RichText
Use the Scribe framework (Qt Rich Text) to render the text.
- MathMLText
Use a MathML (<http://en.wikipedia.org/wiki/MathML>) render engine to display the text. The Qwt MathML extension offers such an engine based on the MathML renderer of the Qt solutions package. Unfortunately it is only available for owners of a commercial Qt license.
- TeXText
Use a TeX (<http://en.wikipedia.org/wiki/TeX>) render engine to display the text.
- OtherFormat
The number of text formats can be extended using `setTextEngine`. Formats \geq OtherFormat are not used by Qwt.

See also:

[QwtTextEngine](#), [setTextEngine](#)

Definition at line 85 of file `qwt_text.h`.

6.83.2.2 enum [QwtText::PaintAttribute](#)

Paint Attributes.

Font and color and background are optional attributes of a [QwtText](#). The paint attributes hold the information, if they are set.

- PaintUsingTextFont
The text has an individual font.
- PaintUsingTextColor
The text has an individual color.
- PaintBackground
The text has an individual background.

Definition at line 111 of file `qwt_text.h`.

6.83.2.3 enum [QwtText::LayoutAttribute](#)

Layout Attributes.

The layout attributes affects some aspects of the layout of the text.

- MinimumLayout
Layout the text without its margins. This mode is useful if a text needs to be aligned accurately, like the tick labels of a scale. If [QwtTextEngine::textMargins](#) is not implemented for the format of the text, MinimumLayout has no effect.

Definition at line 129 of file `qwt_text.h`.

6.83.3 Constructor & Destructor Documentation

6.83.3.1 QwtText::QwtText (const QString & *text* = QString::null, QwtText::TextFormat *textFormat* = AutoText)

Constructor

Parameters:

text Text content
textFormat Text format

Definition at line 180 of file qwt_text.cpp.

References `textEngine()`.

6.83.3.2 QwtText::QwtText (const QwtText &)

Copy constructor.

Definition at line 190 of file qwt_text.cpp.

References `d_data`, and `d_layoutCache`.

6.83.3.3 QwtText::~QwtText ()

Destructor.

Definition at line 200 of file qwt_text.cpp.

6.83.4 Member Function Documentation

6.83.4.1 QwtText & QwtText::operator= (const QwtText &)

Assignment operator.

Definition at line 207 of file qwt_text.cpp.

References `d_data`, and `d_layoutCache`.

6.83.4.2 void QwtText::setText (const QString & *text*, QwtText::TextFormat *textFormat* = AutoText)

Assign a new text content

Parameters:

text Text content
textFormat Text format

Definition at line 237 of file qwt_text.cpp.

References `textEngine()`.

6.83.4.3 QString QwtText::text () const

Return the text.

See also:

[setText](#)

Definition at line 249 of file qwt_text.cpp.

Referenced by QwtPlot::grabProperties().

6.83.4.4 bool QwtText::isNull () const [inline]

Returns:

[text\(\).isNull\(\)](#)

Definition at line 149 of file qwt_text.h.

6.83.4.5 bool QwtText::isEmpty () const [inline]

Returns:

[text\(\).isEmpty\(\)](#)

Definition at line 152 of file qwt_text.h.

Referenced by QwtScaleDraw::boundingLabelRect(), QwtScaleDraw::drawLabel(), QwtRoundScaleDraw::drawLabel(), QwtPicker::drawTracker(), QwtRoundScaleDraw::extent(), QwtScaleDraw::labelRect(), and QwtPicker::trackerRect().

6.83.4.6 void QwtText::setFont (const QFont & font)

Set the font.

Parameters:

font Font

Note:

Setting the font might have no effect, when the text contains control sequences for setting fonts.

Definition at line 289 of file qwt_text.cpp.

References [setPaintAttribute\(\)](#).

Referenced by QwtPlotPrintFilter::apply(), and QwtPicker::drawTracker().

6.83.4.7 QFont QwtText::font () const

Return the font.

Definition at line 296 of file qwt_text.cpp.

Referenced by QwtPlotPrintFilter::apply(), [draw\(\)](#), [heightForWidth\(\)](#), and [textSize\(\)](#).

6.83.4.8 QFont QwtText::usedFont (const QFont & *defaultFont*) const

Return the font of the text, if it has one. Otherwise return defaultFont.

Parameters:

defaultFont Default font

See also:

[setFont](#), [font](#), [PaintAttributes](#)

Definition at line 308 of file qwt_text.cpp.

Referenced by [QwtPicker::drawTracker\(\)](#), [heightForWidth\(\)](#), and [textSize\(\)](#).

6.83.4.9 void QwtText::setRenderFlags (int *renderFlags*)

Change the render flags.

The default setting is Qt::AlignCenter

Parameters:

renderFlags Bitwise OR of the flags used like in [QPainter::drawText](#)

See also:

[renderFlags](#), [QwtTextEngine::draw](#)

Note:

Some renderFlags might have no effect, depending on the text format.

Definition at line 264 of file qwt_text.cpp.

Referenced by [QwtScaleWidget::drawTitle\(\)](#), [QwtLegendItem::setText\(\)](#), [QwtScaleWidget::setTitle\(\)](#), and [QwtAbstractScaleDraw::tickLabel\(\)](#).

6.83.4.10 int QwtText::renderFlags () const**Returns:**

Render flags

See also:

[setRenderFlags](#)

Definition at line 277 of file qwt_text.cpp.

Referenced by [QwtScaleWidget::setTitle\(\)](#).

6.83.4.11 void QwtText::setColor (const QColor & *color*)

Set the pen color used for painting the text.

Parameters:

color Color

Note:

Setting the color might have no effect, when the text contains control sequences for setting colors.

Definition at line 323 of file qwt_text.cpp.

References `setPaintAttribute()`.

Referenced by `QwtPlotPrintFilter::apply()`, and `QwtPlotPrintFilter::reset()`.

6.83.4.12 QColor QwtText::color () const

Return the pen color, used for painting the text.

Definition at line 330 of file qwt_text.cpp.

Referenced by `QwtPlotPrintFilter::apply()`.

6.83.4.13 QColor QwtText::usedColor (const QColor & defaultColor) const

Return the color of the text, if it has one. Otherwise return `defaultColor`.

Parameters:

defaultColor Default color

See also:

[setColor](#), [color](#), [PaintAttributes](#)

Definition at line 342 of file qwt_text.cpp.

6.83.4.14 void QwtText::setBackgroundPen (const QPen & pen)

Set the background pen

Parameters:

pen Background pen

See also:

[backgroundPen](#), [setBackgroundBrush](#)

Definition at line 356 of file qwt_text.cpp.

References `setPaintAttribute()`.

6.83.4.15 QPen QwtText::backgroundPen () const**Returns:**

Background pen

See also:

[setBackgroundPen](#), [backgroundBrush](#)

Definition at line 366 of file qwt_text.cpp.

6.83.4.16 void QwtText::setBackgroundBrush (const QBrush & *brush*)

Set the background brush

Parameters:

brush Background brush

See also:

[backgroundBrush](#), [setBackgroundPen](#)

Definition at line 377 of file qwt_text.cpp.

References [setPaintAttribute\(\)](#).

6.83.4.17 QBrush QwtText::backgroundBrush () const

Returns:

Background brush

See also:

[setBackgroundBrush](#), [backgroundPen](#)

Definition at line 387 of file qwt_text.cpp.

6.83.4.18 void QwtText::setPaintAttribute ([PaintAttribute](#) *attribute*, bool *on* = true)

Change a paint attribute

Parameters:

attribute Paint attribute

on On/Off

Note:

Used by [setFont](#), [setColor](#), [setBackgroundPen](#) and [setBackgroundBrush](#)

See also:

[testPaintAttribute](#)

Definition at line 401 of file qwt_text.cpp.

Referenced by [setBackgroundBrush\(\)](#), [setBackgroundPen\(\)](#), [setColor\(\)](#), and [setFont\(\)](#).

6.83.4.19 `bool QwtText::testPaintAttribute (PaintAttribute attribute) const`

Test a paint attribute

Parameters:

attribute Paint attribute

Returns:

true, if attribute is enabled

See also:

[setPaintAttribute](#)

Definition at line 417 of file qwt_text.cpp.

Referenced by `QwtPlotPrintFilter::apply()`.

6.83.4.20 `void QwtText::setLayoutAttribute (LayoutAttribute attribute, bool on = true)`

Change a layout attribute

Parameters:

attribute Layout attribute

on On/Off

See also:

[testLayoutAttribute](#)

Definition at line 429 of file qwt_text.cpp.

6.83.4.21 `bool QwtText::testLayoutAttribute (LayoutAttribute attribute) const`

Test a layout attribute

Parameters:

attribute Layout attribute

Returns:

true, if attribute is enabled

See also:

[setLayoutAttribute](#)

Definition at line 445 of file qwt_text.cpp.

6.83.4.22 int QwtText::heightForWidth (int *width*, const QFont & *defaultFont* = QFont ()) const

Find the height for a given width

Parameters:

defaultFont Font, used for the calculation if the text has no font

width Width

Returns:

Calculated height

Definition at line 458 of file qwt_text.cpp.

References font(), QwtMetricsMap::layoutToScreenX(), QwtPainter::metricsMap(), QwtMetricsMap::screenToLayoutY(), and usedFont().

6.83.4.23 QSize QwtText::textSize (const QFont & *defaultFont* = QFont ()) const

Returns the size, that is needed to render text

Parameters:

defaultFont Font of the text

Returns:

Calculated size

Definition at line 510 of file qwt_text.cpp.

References font(), QwtMetricsMap::isIdentity(), QwtPainter::metricsMap(), QwtMetricsMap::screenToLayout(), and usedFont().

Referenced by QwtScaleDraw::boundingLabelRect(), QwtScaleDraw::drawLabel(), QwtRoundScaleDraw::drawLabel(), QwtRoundScaleDraw::extent(), QwtScaleDraw::labelRect(), and QwtPicker::trackerRect().

6.83.4.24 void QwtText::draw (QPainter * *painter*, const QRect & *rect*) const

Draw a text into a rectangle

Parameters:

painter Painter

rect Rectangle

Definition at line 564 of file qwt_text.cpp.

References QwtPainter::drawRect(), font(), QwtPainter::metricsMap(), QwtMetricsMap::screenToLayoutX(), and QwtMetricsMap::screenToLayoutY().

Referenced by QwtLegendItem::drawItem(), QwtScaleDraw::drawLabel(), QwtRoundScaleDraw::drawLabel(), QwtScaleWidget::drawTitle(), QwtPicker::drawTracker(), and QwtPlot::printTitle().

6.83.4.25 `const QwtTextEngine * QwtText::textEngine (const QString & text, QwtText::TextFormat format = AutoText) [static]`

Find the text engine for a text format

In case of `QwtText::AutoText` the first text engine (beside `QwtPlainTextEngine`) is returned, where `QwtTextEngine::mightRender` returns true. If there is none `QwtPlainTextEngine` is returned.

If no text engine is registered for the format `QwtPlainTextEngine` is returned.

Parameters:

text Text, needed in case of `AutoText`

format Text format

Definition at line 646 of file `qwt_text.cpp`.

Referenced by `QwtText()`, and `setText()`.

6.83.4.26 `const QwtTextEngine * QwtText::textEngine (QwtText::TextFormat format) [static]`

Find the text engine for a text format.

`textEngine` can be used to find out if a text format is supported. F.e, if one wants to use MathML labels, the MathML renderer from the commercial Qt solutions package might be required, that is not available in Qt Open Source Edition environments.

Parameters:

format Text format

Returns:

The text engine, or NULL if no engine is available.

Definition at line 701 of file `qwt_text.cpp`.

6.83.4.27 `void QwtText::setTextEngine (QwtText::TextFormat format, QwtTextEngine * engine) [static]`

Assign/Replace a text engine for a text format

With `setTextEngine` it is possible to extend Qwt with other types of text formats.

Owner of a commercial Qt license can build the `qwtmathml` library, that is based on the MathML renderer, that is included in MML Widget component of the Qt solutions package.

For `QwtText::PlainText` it is not allowed to assign a engine == NULL.

Parameters:

format Text format

engine Text engine

See also:

[QwtMathMLTextEngine](#)

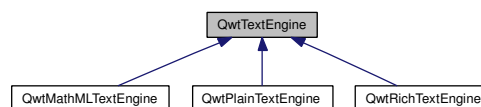
Warning:

Using `QwtText::AutoText` does nothing.

Definition at line 681 of file `qwt_text.cpp`.

6.84 QwtTextEngine Class Reference

Inheritance diagram for `QwtTextEngine`:

**6.84.1 Detailed Description**

Abstract base class for rendering text strings.

A text engine is responsible for rendering texts for a specific text format. They are used by `QwtText` to render a text.

`QwtPlainTextEngine` and `QwtRichTextEngine` are part of the Qwt library.

`QwtMathMLTextEngine` can be found in Qwt MathML extension, that needs the MathML renderer of the Qt solutions package. Unfortunately it is only available with a commercial Qt license.

See also:

`QwtText::setTextEngine`

Definition at line 38 of file `qwt_text_engine.h`.

Public Member Functions

- virtual `~QwtTextEngine()`
- virtual `int heightForWidth(const QFont &font, int flags, const QString &text, int width) const=0`
- virtual `QSize textSize(const QFont &font, int flags, const QString &text) const=0`
- virtual `bool mightRender(const QString &text) const=0`
- virtual `void textMargins(const QFont &font, const QString &text, int &left, int &right, int &top, int &bottom) const=0`
- virtual `void draw(QPainter *painter, const QRect &rect, int flags, const QString &text) const=0`

Protected Member Functions

- `QwtTextEngine()`

6.84.2 Constructor & Destructor Documentation**6.84.2.1 QwtTextEngine::~QwtTextEngine() [virtual]**

Destructor.

Definition at line 175 of file `qwt_text_engine.cpp`.

6.84.2.2 QwtTextEngine::QwtTextEngine () [protected]

Constructor.

Definition at line 170 of file qwt_text_engine.cpp.

6.84.3 Member Function Documentation

6.84.3.1 virtual int QwtTextEngine::heightForWidth (const QFont & *font*, int *flags*, const QString & *text*, int *width*) const [pure virtual]

Find the height for a given width

Parameters:

font Font of the text

flags Bitwise OR of the flags used like in QPainter::drawText

text Text to be rendered

width Width

Returns:

Calculated height

Implemented in [QwtPlainTextEngine](#), [QwtRichTextEngine](#), and [QwtMathMLTextEngine](#).

6.84.3.2 virtual QSize QwtTextEngine::textSize (const QFont & *font*, int *flags*, const QString & *text*) const [pure virtual]

Returns the size, that is needed to render text

Parameters:

font Font of the text

flags Bitwise OR of the flags like in for QPainter::drawText

text Text to be rendered

Returns:

Calculated size

Implemented in [QwtPlainTextEngine](#), [QwtRichTextEngine](#), and [QwtMathMLTextEngine](#).

6.84.3.3 virtual bool QwtTextEngine::mightRender (const QString & *text*) const [pure virtual]

Test if a string can be rendered by this text engine

Parameters:

text Text to be tested

Returns:

true, if it can be rendered

Implemented in [QwtPlainTextEngine](#), [QwtRichTextEngine](#), and [QwtMathMLTextEngine](#).

6.84.3.4 `virtual void QwtTextEngine::textMargins (const QFont & font, const QString & text, int & left, int & right, int & top, int & bottom) const` [pure virtual]

Return margins around the texts

The `textSize` might include margins around the text, like `QFontMetrics::descent`. In situations where texts need to be aligned in detail, knowing these margins might improve the layout calculations.

Parameters:

font Font of the text
text Text to be rendered
left Return value for the left margin
right Return value for the right margin
top Return value for the top margin
bottom Return value for the bottom margin

Implemented in [QwtPlainTextEngine](#), [QwtRichTextEngine](#), and [QwtMathMLTextEngine](#).

6.84.3.5 `virtual void QwtTextEngine::draw (QPainter * painter, const QRect & rect, int flags, const QString & text) const` [pure virtual]

Draw the text in a clipping rectangle

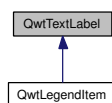
Parameters:

painter Painter
rect Clipping rectangle
flags Bitwise OR of the flags like in for `QPainter::drawText`
text Text to be rendered

Implemented in [QwtPlainTextEngine](#), [QwtRichTextEngine](#), and [QwtMathMLTextEngine](#).

6.85 QwtTextLabel Class Reference

Inheritance diagram for QwtTextLabel:



6.85.1 Detailed Description

A Widget which displays a [QwtText](#).

Definition at line 25 of file `qwt_text_label.h`.

Public Slots

- void [setText](#) (const QString &, [QwtText::TextFormat](#) textFormat=[QwtText::AutoText](#))
- virtual void [setText](#) (const [QwtText](#) &)
- void [clear](#) ()

Public Member Functions

- [QwtTextLabel](#) (QWidget *parent=NULL)
- [QwtTextLabel](#) (const [QwtText](#) &, QWidget *parent=NULL)
- virtual [~QwtTextLabel](#) ()
- const [QwtText](#) & [text](#) () const
- int [indent](#) () const
- void [setIndent](#) (int)
- int [margin](#) () const
- void [setMargin](#) (int)
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- virtual int [heightForWidth](#) (int) const
- QRect [textRect](#) () const

Protected Member Functions

- virtual void [paintEvent](#) (QPaintEvent *e)
- virtual void [drawContents](#) (QPainter *)
- virtual void [drawText](#) (QPainter *, const QRect &)

6.85.2 Constructor & Destructor Documentation**6.85.2.1 QwtTextLabel::QwtTextLabel (QWidget *parent = NULL) [explicit]**

Constructs an empty label.

Parameters:

parent Parent widget

Definition at line 36 of file qwt_text_label.cpp.

6.85.2.2 QwtTextLabel::QwtTextLabel (const [QwtText](#) & text, QWidget *parent = NULL) [explicit]

Constructs a label that displays the text, text

Parameters:

parent Parent widget

text Text

Definition at line 60 of file qwt_text_label.cpp.

References [text\(\)](#).

6.85.2.3 QwtTextLabel::~QwtTextLabel () [virtual]

Destructor.

Definition at line 68 of file qwt_text_label.cpp.

6.85.3 Member Function Documentation

6.85.3.1 `void QwtTextLabel::setText (const QString & text, QwtText::TextFormat textFormat = QwtText::AutoText)` [slot]

Change the label's text, keeping all other [QwtText](#) attributes

Parameters:

text New text

textFormat Format of text

See also:

[QwtText](#)

Definition at line 86 of file qwt_text_label.cpp.

Referenced by [QwtLegendItem::setText\(\)](#).

6.85.3.2 `void QwtTextLabel::setText (const QwtText & text)` [virtual, slot]

Change the label's text

Parameters:

text New text

Reimplemented in [QwtLegendItem](#).

Definition at line 98 of file qwt_text_label.cpp.

References [text\(\)](#).

6.85.3.3 `void QwtTextLabel::clear ()` [slot]

Clear the text and all [QwtText](#) attributes.

Definition at line 113 of file qwt_text_label.cpp.

6.85.3.4 `const QwtText & QwtTextLabel::text () const`

Return the text.

Definition at line 107 of file qwt_text_label.cpp.

Referenced by [QwtLegendItem::drawItem\(\)](#), [QwtPlot::printTitle\(\)](#), [QwtLegendItem::QwtLegendItem\(\)](#), [QwtTextLabel\(\)](#), [QwtPlotPrintFilter::reset\(\)](#), [setText\(\)](#), and [QwtLegendItem::setText\(\)](#).

6.85.3.5 `int QwtTextLabel::indent () const`

Return label's text indent in pixels.

Definition at line 122 of file qwt_text_label.cpp.

Referenced by [heightForWidth\(\)](#), [minimumSizeHint\(\)](#), and [textRect\(\)](#).

6.85.3.6 void QwtTextLabel::setIndent (int *indent*)

Set label's text indent in pixels

Parameters:

indent Indentation in pixels

Definition at line 131 of file qwt_text_label.cpp.

Referenced by QwtLegendItem::setIdentifierWidth(), and QwtLegendItem::setSpacing().

6.85.3.7 int QwtTextLabel::margin () const

Return label's text indent in pixels.

Definition at line 143 of file qwt_text_label.cpp.

Referenced by drawContents(), QwtLegendItem::drawItem(), QwtLegendItem::paintEvent(), QwtLegendItem::setIdentifierWidth(), and QwtLegendItem::setSpacing().

6.85.3.8 void QwtTextLabel::setMargin (int *margin*)

Set label's margin in pixels

Parameters:

margin Margin in pixels

Definition at line 152 of file qwt_text_label.cpp.

Referenced by QwtLegendItem::setItemMode().

6.85.3.9 QSize QwtTextLabel::sizeHint () const [virtual]

Return label's margin in pixels.

Reimplemented in [QwtLegendItem](#).

Definition at line 161 of file qwt_text_label.cpp.

References minimumSizeHint().

Referenced by QwtLegendItem::sizeHint().

6.85.3.10 QSize QwtTextLabel::minimumSizeHint () const [virtual]

Return a minimum size hint.

Definition at line 167 of file qwt_text_label.cpp.

References indent().

Referenced by sizeHint().

6.85.3.11 int QwtTextLabel::heightForWidth (int *width*) const [virtual]

Returns the preferred height for this widget, given the width.

Parameters:

width Width

Definition at line 196 of file qwt_text_label.cpp.

References `indent()`.

6.85.3.12 `QRect QwtTextLabel::textRect () const`

Calculate the rect for the text in widget coordinates

Returns:

Text rect

Definition at line 279 of file qwt_text_label.cpp.

References `indent()`.

Referenced by `drawContents()`.

6.85.3.13 `void QwtTextLabel::paintEvent (QPaintEvent * e) [protected, virtual]`

Qt paint event.

Reimplemented in [QwtLegendItem](#).

Definition at line 218 of file qwt_text_label.cpp.

References `drawContents()`.

6.85.3.14 `void QwtTextLabel::drawContents (QPainter *) [protected, virtual]`

Redraw the text and focus indicator.

Definition at line 241 of file qwt_text_label.cpp.

References `QwtPainter::drawFocusRect()`, `drawText()`, `margin()`, and `textRect()`.

Referenced by `paintEvent()`, and `QwtLegendItem::paintEvent()`.

6.85.3.15 `void QwtTextLabel::drawText (QPainter *, const QRect &) [protected, virtual]`

Redraw the text.

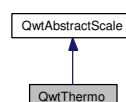
Reimplemented in [QwtLegendItem](#).

Definition at line 270 of file qwt_text_label.cpp.

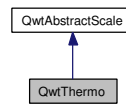
Referenced by `drawContents()`, and `QwtLegendItem::drawText()`.

6.86 QwtThermo Class Reference

Inheritance diagram for QwtThermo:



Collaboration diagram for QwtThermo:



6.86.1 Detailed Description

The Thermometer Widget.

[QwtThermo](#) is a widget which displays a value in an interval. It supports:

- a horizontal or vertical layout;
- a range;
- a scale;
- an alarm level.

By default, the scale and range run over the same interval of values. [QwtAbstractScale::setScale\(\)](#) changes the interval of the scale and allows easy conversion between physical units.

The example shows how to make the scale indicate in degrees Fahrenheit and to set the value in degrees Kelvin:

```
#include <qapplication.h>
#include <qwt_thermo.h>

double Kelvin2Fahrenheit(double kelvin)
{
    // see http://en.wikipedia.org/wiki/Kelvin
    return 1.8*kelvin - 459.67;
}

int main(int argc, char **argv)
{
    const double minKelvin = 0.0;
    const double maxKelvin = 500.0;

    QApplication a(argc, argv);
    QwtThermo t;
    t.setRange(minKelvin, maxKelvin);
    t.setScale(Kelvin2Fahrenheit(minKelvin), Kelvin2Fahrenheit(maxKelvin));
    // set the value in Kelvin but the scale displays in Fahrenheit
    // 273.15 Kelvin = 0 Celsius = 32 Fahrenheit
    t.setValue(273.15);
    a.setMainWidget(&t);
    t.show();
    return a.exec();
}
```

Todo

Improve the support for a logarithmic range and/or scale.

Definition at line 69 of file qwt_thermo.h.

Public Types

- enum [ScalePos](#) {
 NoScale,
 LeftScale,
 RightScale,
 TopScale,
 BottomScale,
 NoScale,
 LeftScale,
 RightScale,
 TopScale,
 BottomScale }

Public Slots

- void [setValue](#) (double val)

Public Member Functions

- [QwtThermo](#) (QWidget *parent=NULL)
- virtual [~QwtThermo](#) ()
- void [setOrientation](#) (Qt::Orientation o, [ScalePos](#) s)
- void [setScalePosition](#) ([ScalePos](#) s)
- [ScalePos](#) [scalePosition](#) () const
- void [setBorderWidth](#) (int w)
- int [borderWidth](#) () const
- void [setFillBrush](#) (const QBrush &b)
- const QBrush & [fillBrush](#) () const
- void [setFillColor](#) (const QColor &c)
- const QColor & [fillColor](#) () const
- void [setAlarmBrush](#) (const QBrush &b)
- const QBrush & [alarmBrush](#) () const
- void [setAlarmColor](#) (const QColor &c)
- const QColor & [alarmColor](#) () const
- void [setAlarmLevel](#) (double v)
- double [alarmLevel](#) () const
- void [setAlarmEnabled](#) (bool tf)
- bool [alarmEnabled](#) () const
- void [setPipeWidth](#) (int w)
- int [pipeWidth](#) () const
- void [setMaxValue](#) (double v)
- double [maxValue](#) () const
- void [setMinValue](#) (double v)
- double [minValue](#) () const
- double [value](#) () const
- void [setRange](#) (double vmin, double vmax, bool lg=false)

- void [setMargin](#) (int m)
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const
- void [setScaleDraw](#) (QwtScaleDraw *)
- const QwtScaleDraw * [scaleDraw](#) () const

Protected Member Functions

- void [draw](#) (QPainter *p, const QRect &update_rect)
- void [drawThermo](#) (QPainter *p)
- void [layoutThermo](#) (bool update=true)
- virtual void [scaleChange](#) ()
- virtual void [fontChange](#) (const QFont &oldFont)
- virtual void [paintEvent](#) (QPaintEvent *e)
- virtual void [resizeEvent](#) (QResizeEvent *e)
- QwtScaleDraw * [scaleDraw](#) ()

6.86.2 Constructor & Destructor Documentation

6.86.2.1 QwtThermo::QwtThermo (QWidget * *parent* = NULL) [explicit]

Constructor

Parameters:

parent Parent widget

Definition at line 64 of file qwt_thermo.cpp.

6.86.2.2 QwtThermo::~QwtThermo () [virtual]

Destructor.

Definition at line 105 of file qwt_thermo.cpp.

6.86.3 Member Function Documentation

6.86.3.1 void QwtThermo::setOrientation (Qt::Orientation *o*, [ScalePos](#) *s*)

Set the thermometer orientation and the scale position.

The scale position NoScale disables the scale.

Parameters:

o orientation. Possible values are Qt::Horizontal and Qt::Vertical. The default value is Qt::Vertical.

s Position of the scale. The default value is NoScale.

A valid combination of scale position and orientation is enforced:

- a horizontal thermometer can have the scale positions TopScale, BottomScale or NoScale;
- a vertical thermometer can have the scale positions LeftScale, RightScale or NoScale;

- an invalid scale position will default to NoScale.

See also:

[QwtThermo::setScalePosition\(\)](#)

Definition at line 365 of file qwt_thermo.cpp.

References [layoutThermo\(\)](#).

Referenced by [setScalePosition\(\)](#).

6.86.3.2 void QwtThermo::setScalePosition (ScalePos s)

Change the scale position (and thermometer orientation).

Parameters:

s Position of the scale.

A valid combination of scale position and orientation is enforced:

- if the new scale position is LeftScale or RightScale, the scale orientation will become Qt::Vertical;
- if the new scale position is BottomScale or TopScale, the scale orientation will become Qt::Horizontal;
- if the new scale position is NoScale, the scale orientation will not change.

See also:

[QwtThermo::setOrientation\(\)](#)

Definition at line 428 of file qwt_thermo.cpp.

References [setOrientation\(\)](#).

6.86.3.3 QwtThermo::ScalePos QwtThermo::scalePosition () const

Return the scale position.

Definition at line 439 of file qwt_thermo.cpp.

6.86.3.4 void QwtThermo::setBorderWidth (int w)

Set the border width of the pipe.

Definition at line 623 of file qwt_thermo.cpp.

References [layoutThermo\(\)](#).

6.86.3.5 int QwtThermo::borderWidth () const

Return the border width of the thermometer pipe.

Definition at line 634 of file qwt_thermo.cpp.

6.86.3.6 void QwtThermo::setFillBrush (const QBrush & *brush*)

Change the brush of the liquid.

Parameters:

brush New brush. The default brush is solid black.

Definition at line 674 of file qwt_thermo.cpp.

6.86.3.7 const QBrush & QwtThermo::fillBrush () const

Return the liquid brush.

Definition at line 681 of file qwt_thermo.cpp.

6.86.3.8 void QwtThermo::setFillColor (const QColor & *c*)

Change the color of the liquid.

Parameters:

c New color. The default color is black.

Definition at line 690 of file qwt_thermo.cpp.

6.86.3.9 const QColor & QwtThermo::fillColor () const

Return the liquid color.

Definition at line 697 of file qwt_thermo.cpp.

6.86.3.10 void QwtThermo::setAlarmBrush (const QBrush & *brush*)

Specify the liquid brush above the alarm threshold.

Parameters:

brush New brush. The default is solid white.

Definition at line 706 of file qwt_thermo.cpp.

6.86.3.11 const QBrush & QwtThermo::alarmBrush () const

Return the liquid brush above the alarm threshold.

Definition at line 713 of file qwt_thermo.cpp.

6.86.3.12 void QwtThermo::setAlarmColor (const QColor & *c*)

Specify the liquid color above the alarm threshold.

Parameters:

c New color. The default is white.

Definition at line 722 of file qwt_thermo.cpp.

6.86.3.13 const QColor & QwtThermo::alarmColor () const

Return the liquid color above the alarm threshold.

Definition at line 729 of file qwt_thermo.cpp.

6.86.3.14 void QwtThermo::setAlarmLevel (double *v*)

Specify the alarm threshold.

Definition at line 735 of file qwt_thermo.cpp.

6.86.3.15 double QwtThermo::alarmLevel () const

Return the alarm threshold.

Definition at line 743 of file qwt_thermo.cpp.

6.86.3.16 void QwtThermo::setAlarmEnabled (bool *tf*)

Enable or disable the alarm threshold.

Parameters:

tf true (disabled) or false (enabled)

Definition at line 788 of file qwt_thermo.cpp.

6.86.3.17 bool QwtThermo::alarmEnabled () const

Return if the alarm threshold is enabled or disabled.

Definition at line 795 of file qwt_thermo.cpp.

6.86.3.18 void QwtThermo::setPipeWidth (int *w*)

Change the width of the pipe.

Definition at line 749 of file qwt_thermo.cpp.

References layoutThermo().

6.86.3.19 int QwtThermo::pipeWidth () const

Return the width of the pipe.

Definition at line 759 of file qwt_thermo.cpp.

6.86.3.20 void QwtThermo::setMaxValue (double *v*)

Set the maximum value.

Definition at line 111 of file qwt_thermo.cpp.

References setRange().

6.86.3.21 double QwtThermo::maxValue () const

Return the maximum value.

Definition at line 117 of file qwt_thermo.cpp.

6.86.3.22 void QwtThermo::setMinValue (double v)

Set the minimum value.

Definition at line 123 of file qwt_thermo.cpp.

References `setRange()`.

6.86.3.23 double QwtThermo::minValue () const

Return the minimum value.

Definition at line 129 of file qwt_thermo.cpp.

6.86.3.24 double QwtThermo::value () const

Return the value.

Definition at line 145 of file qwt_thermo.cpp.

6.86.3.25 void QwtThermo::setRange (double vmin, double vmax, bool logarithmic = false)

Set the range.

Parameters:

- vmin* value corresponding lower or left end of the thermometer
- vmax* value corresponding to the upper or right end of the thermometer
- logarithmic* logarithmic mapping, true or false

Definition at line 645 of file qwt_thermo.cpp.

References `QwtAbstractScale::autoScale()`, `layoutThermo()`, `QwtAbstractScale::rescale()`, `QwtAbstractScale::scaleEngine()`, and `QwtAbstractScale::setScaleEngine()`.

Referenced by `setMaxValue()`, and `setMinValue()`.

6.86.3.26 void QwtThermo::setMargin (int m)

Specify the distance between the pipe's endpoints and the widget's border.

The margin is used to leave some space for the scale labels. If a large font is used, it is advisable to adjust the margins.

Parameters:

- m* New Margin. The default values are 10 for horizontal orientation and 20 for vertical orientation.

Warning:

- The margin has no effect if the scale is disabled.
- This function is a NOOP because margins are determined automatically.

Definition at line 779 of file qwt_thermo.cpp.

6.86.3.27 `QSize QwtThermo::sizeHint () const` [virtual]**Returns:**

the minimum size hint

See also:

[QwtThermo::minimumSizeHint](#)

Definition at line 804 of file `qwt_thermo.cpp`.

References `minimumSizeHint()`.

6.86.3.28 `QSize QwtThermo::minimumSizeHint () const` [virtual]

Return a minimum size hint.

Warning:

The return value depends on the font and the scale.

See also:

[QwtThermo::sizeHint](#)

Definition at line 814 of file `qwt_thermo.cpp`.

References `QwtScaleDraw::extent()`, `QwtScaleDraw::minLength()`, and `scaleDraw()`.

Referenced by `sizeHint()`.

6.86.3.29 `void QwtThermo::setScaleDraw (QwtScaleDraw * scaleDraw)`

Set a scale draw.

For changing the labels of the scales, it is necessary to derive from [QwtScaleDraw](#) and overload [QwtScaleDraw::label\(\)](#).

Parameters:

scaleDraw ScaleDraw object, that has to be created with `new` and will be deleted in `~QwtThermo` or the next call of [setScaleDraw\(\)](#).

Definition at line 161 of file `qwt_thermo.cpp`.

References `scaleDraw()`, and `QwtAbstractScale::setAbstractScaleDraw()`.

6.86.3.30 `const QwtScaleDraw * QwtThermo::scaleDraw () const`**Returns:**

the scale draw of the thermo

See also:

[setScaleDraw\(\)](#)

Definition at line 170 of file `qwt_thermo.cpp`.

References `QwtAbstractScale::abstractScaleDraw()`.

Referenced by `draw()`, `layoutThermo()`, `minimumSizeHint()`, and `setScaleDraw()`.

6.86.3.31 void QwtThermo::setValue (double *val*) [slot]

Set the current value.

Definition at line 135 of file qwt_thermo.cpp.

6.86.3.32 void QwtThermo::draw (QPainter **p*, const QRect & *update_rect*) [protected]

Draw the whole [QwtThermo](#).

Definition at line 202 of file qwt_thermo.cpp.

References [QwtAbstractScaleDraw::draw\(\)](#), [drawThermo\(\)](#), and [scaleDraw\(\)](#).

Referenced by [paintEvent\(\)](#).

6.86.3.33 void QwtThermo::drawThermo (QPainter **p*) [protected]

Redraw the liquid in thermometer pipe.

Definition at line 459 of file qwt_thermo.cpp.

Referenced by [draw\(\)](#).

6.86.3.34 void QwtThermo::layoutThermo (bool *update_geometry* = true) [protected]

Recalculate the [QwtThermo](#) geometry and layout based on the [QwtThermo::rect\(\)](#) and the fonts.

Parameters:

update_geometry notify the layout system and call update to redraw the scale

Definition at line 242 of file qwt_thermo.cpp.

References [QwtScaleDraw::getBorderDistHint\(\)](#), [QwtScaleDraw::move\(\)](#), [scaleDraw\(\)](#), [QwtScaleDraw::setAlignment\(\)](#), and [QwtScaleDraw::setLength\(\)](#).

Referenced by [fontChange\(\)](#), [resizeEvent\(\)](#), [scaleChange\(\)](#), [setBorderWidth\(\)](#), [setOrientation\(\)](#), [setPipeWidth\(\)](#), and [setRange\(\)](#).

6.86.3.35 void QwtThermo::scaleChange () [protected, virtual]

Notify a scale change.

Reimplemented from [QwtAbstractScale](#).

Definition at line 452 of file qwt_thermo.cpp.

References [layoutThermo\(\)](#).

6.86.3.36 void QwtThermo::fontChange (const QFont & *oldFont*) [protected, virtual]

Notify a font change.

Definition at line 445 of file qwt_thermo.cpp.

References [layoutThermo\(\)](#).

6.86.3.37 void QwtThermo::paintEvent (QPaintEvent **e*) [protected, virtual]

Qt paint event.

Definition at line 185 of file qwt_thermo.cpp.

References `draw()`.

6.86.3.38 `void QwtThermo::resizeEvent (QResizeEvent * e)` [protected, virtual]

Qt resize event handler.

Definition at line 231 of file qwt_thermo.cpp.

References `layoutThermo()`.

6.86.3.39 `QwtScaleDraw * QwtThermo::scaleDraw ()` [protected]

Returns:

the scale draw of the thermo

See also:

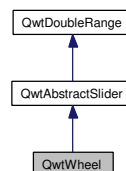
[setScaleDraw\(\)](#)

Definition at line 179 of file qwt_thermo.cpp.

References `QwtAbstractScale::abstractScaleDraw()`.

6.87 QwtWheel Class Reference

Inheritance diagram for QwtWheel:



Collaboration diagram for QwtWheel:



6.87.1 Detailed Description

The Wheel Widget.

The wheel widget can be used to change values over a very large range in very small steps. Using the `setMass` member, it can be configured as a flywheel.

See also:

The radio example.

Definition at line 25 of file qwt_wheel.h.

Public Member Functions

- [QwtWheel](#) (QWidget *parent=NULL)
- virtual [~QwtWheel](#) ()
- virtual void [setOrientation](#) (Qt::Orientation)
- double [totalAngle](#) () const
- double [viewAngle](#) () const
- int [tickCnt](#) () const
- int [internalBorder](#) () const
- double [mass](#) () const
- void [setTotalAngle](#) (double angle)
- void [setTickCnt](#) (int cnt)
- void [setViewAngle](#) (double angle)
- void [setInternalBorder](#) (int width)
- void [setMass](#) (double val)
- void [setWheelWidth](#) (int w)
- virtual QSize [sizeHint](#) () const
- virtual QSize [minimumSizeHint](#) () const

Protected Member Functions

- virtual void [resizeEvent](#) (QResizeEvent *e)
- virtual void [paintEvent](#) (QPaintEvent *e)
- void [layoutWheel](#) (bool update=true)
- void [draw](#) (QPainter *p, const QRect &update_rect)
- void [drawWheel](#) (QPainter *p, const QRect &r)
- void [drawWheelBackground](#) (QPainter *p, const QRect &r)
- void [setColorArray](#) ()
- virtual void [valueChange](#) ()
- virtual void [paletteChange](#) (const QPalette &)
- virtual double [getValue](#) (const QPoint &p)
- virtual void [getScrollMode](#) (const QPoint &p, int &scrollMode, int &direction)

6.87.2 Constructor & Destructor Documentation**6.87.2.1 QwtWheel::QwtWheel (QWidget *parent = NULL) [explicit]**

Constructor.

Definition at line 51 of file qwt_wheel.cpp.

6.87.2.2 QwtWheel::~QwtWheel () [virtual]

Destructor.

Definition at line 86 of file qwt_wheel.cpp.

6.87.3 Member Function Documentation

6.87.3.1 void QwtWheel::setOrientation (Qt::Orientation *o*) [virtual]

Set the wheel's orientation.

Parameters:

o Orientation. Allowed values are Qt::Horizontal and Qt::Vertical. Defaults to Qt::Horizontal.

See also:

[QwtAbstractSlider::orientation\(\)](#)

Reimplemented from [QwtAbstractSlider](#).

Definition at line 325 of file qwt_wheel.cpp.

References [layoutWheel\(\)](#), [QwtAbstractSlider::orientation\(\)](#), and [QwtAbstractSlider::setOrientation\(\)](#).

6.87.3.2 double QwtWheel::mass () const [virtual]

Returns:

mass

Reimplemented from [QwtAbstractSlider](#).

Definition at line 168 of file qwt_wheel.cpp.

References [QwtAbstractSlider::mass\(\)](#).

6.87.3.3 void QwtWheel::setTotalAngle (double *angle*)

Set the total angle which the wheel can be turned.

One full turn of the wheel corresponds to an angle of 360 degrees. A total angle of $n \times 360$ degrees means that the wheel has to be turned n times around its axis to get from the minimum value to the maximum value.

The default setting of the total angle is 360 degrees.

Parameters:

angle total angle in degrees

Definition at line 304 of file qwt_wheel.cpp.

6.87.3.4 void QwtWheel::setTickCnt (int *cnt*)

Adjust the number of grooves in the wheel's surface.

The number of grooves is limited to $6 \leq cnt \leq 50$. Values outside this range will be clipped. The default value is 10.

Parameters:

cnt Number of grooves per 360 degrees

Definition at line 154 of file qwt_wheel.cpp.

6.87.3.5 void QwtWheel::setViewAngle (double *angle*)

Specify the visible portion of the wheel.

You may use this function for fine-tuning the appearance of the wheel. The default value is 175 degrees. The value is limited from 10 to 175 degrees.

Parameters:

angle Visible angle in degrees

Definition at line 359 of file qwt_wheel.cpp.

6.87.3.6 void QwtWheel::setInternalBorder (int *w*)

Set the internal border width of the wheel.

The internal border must not be smaller than 1 and is limited in dependence on the wheel's size. Values outside the allowed range will be clipped.

The internal border defaults to 2.

Parameters:

w border width

Definition at line 183 of file qwt_wheel.cpp.

References `layoutWheel()`.

6.87.3.7 void QwtWheel::setMass (double *val*) [virtual]

Set the mass of the wheel.

Assigning a mass turns the wheel into a flywheel.

Parameters:

val the wheel's mass

Reimplemented from [QwtAbstractSlider](#).

Definition at line 612 of file qwt_wheel.cpp.

References `QwtAbstractSlider::setMass()`.

6.87.3.8 void QwtWheel::setWheelWidth (int *w*)

Set the width of the wheel.

Corresponds to the wheel height for horizontal orientation, and the wheel width for vertical orientation.

Parameters:

w the wheel's width

Definition at line 624 of file qwt_wheel.cpp.

References `layoutWheel()`.

6.87.3.9 QSize QwtWheel::sizeHint () const [virtual]**Returns:**

a size hint

Definition at line 633 of file qwt_wheel.cpp.

References `minimumSizeHint()`.

6.87.3.10 QSize QwtWheel::minimumSizeHint () const [virtual]

Return a minimum size hint.

Warning:

The return value is based on the wheel width.

Definition at line 642 of file qwt_wheel.cpp.

References `QwtAbstractSlider::orientation()`.

Referenced by `sizeHint()`.

6.87.3.11 void QwtWheel::resizeEvent (QResizeEvent * e) [protected, virtual]

Qt Resize Event.

Definition at line 524 of file qwt_wheel.cpp.

References `layoutWheel()`.

6.87.3.12 void QwtWheel::paintEvent (QPaintEvent * e) [protected, virtual]

Qt Paint Event.

Definition at line 547 of file qwt_wheel.cpp.

References `draw()`.

6.87.3.13 void QwtWheel::layoutWheel (bool *update* = true) [protected]

Recalculate the slider's geometry and layout based on.

Definition at line 533 of file qwt_wheel.cpp.

Referenced by `resizeEvent()`, `setInternalBorder()`, `setOrientation()`, and `setWheelWidth()`.

6.87.3.14 void QwtWheel::draw (QPainter * p, const QRect & *update_rect*) [protected]

Redraw panel and wheel.

Definition at line 564 of file qwt_wheel.cpp.

References `QwtPainter::drawFocusRect()`, and `drawWheel()`.

Referenced by `paintEvent()`.

6.87.3.15 void QwtWheel::drawWheel (QPainter **p*, const QRect & *r*) [protected]

Redraw the wheel.

Parameters:

p painter

r contents rectangle

Definition at line 375 of file qwt_wheel.cpp.

References [drawWheelBackground\(\)](#), [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), [QwtAbstractSlider::orientation\(\)](#), and [QwtDoubleRange::value\(\)](#).

Referenced by [draw\(\)](#).

6.87.3.16 void QwtWheel::drawWheelBackground (QPainter * *p*, const QRect & *r*) [protected]

Draw the Wheel's background gradient.

Definition at line 197 of file qwt_wheel.cpp.

References [QwtAbstractSlider::orientation\(\)](#), and [setColorArray\(\)](#).

Referenced by [drawWheel\(\)](#).

6.87.3.17 void QwtWheel::setColorArray () [protected]

Set up the color array for the background pixmap.

Definition at line 96 of file qwt_wheel.cpp.

Referenced by [drawWheelBackground\(\)](#).

6.87.3.18 void QwtWheel::valueChange () [protected, virtual]

Notify value change.

Reimplemented from [QwtAbstractSlider](#).

Definition at line 582 of file qwt_wheel.cpp.

References [QwtAbstractSlider::valueChange\(\)](#).

6.87.3.19 void QwtWheel::paletteChange (const QPalette &) [protected, virtual]

Call [update\(\)](#) when the palette changes.

Definition at line 654 of file qwt_wheel.cpp.

6.87.3.20 double QwtWheel::getValue (const QPoint & *p*) [protected, virtual]

Determine the value corresponding to a specified point.

Implements [QwtAbstractSlider](#).

Definition at line 494 of file qwt_wheel.cpp.

References [QwtDoubleRange::maxValue\(\)](#), [QwtDoubleRange::minValue\(\)](#), and [QwtAbstractSlider::orientation\(\)](#).

6.87.3.21 `void QwtWheel::getScrollMode (const QPoint & p, int & scrollMode, int & direction)`
`[protected, virtual]`

Determine the scrolling mode and direction corresponding to a specified point.

Parameters:

p point
scrollMode scrolling mode
direction direction

Implements [QwtAbstractSlider](#).

Definition at line 596 of file `qwt_wheel.cpp`.

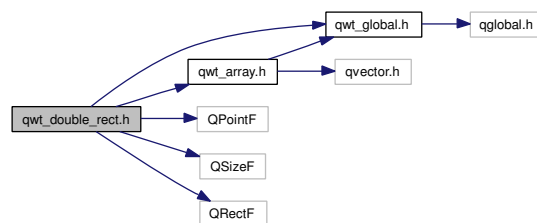
7 Qwt User's Guide File Documentation

7.1 qwt_double_rect.h File Reference

7.1.1 Detailed Description

Definition in file [qwt_double_rect.h](#).

Include dependency graph for `qwt_double_rect.h`:



Defines

- `#define` [QWT_DOUBLE_RECT_H](#) 1

Typedefs

- `typedef` `QPointF` [QwtDoublePoint](#)
- `typedef` `QSizeF` [QwtDoubleSize](#)
- `typedef` `QRectF` [QwtDoubleRect](#)

7.1.2 Typedef Documentation

7.1.2.1 `QPointF` [QwtDoublePoint](#)

This is a typedef, see Trolltech Documentation for `QPointF` in QT assistant 4.x. As soon as Qt3 compatibility is dropped this typedef will disappear.

Definition at line 29 of file `qwt_double_rect.h`.

7.1.2.2 QRectF QwtDoubleRect

This is a typedef, see Trolltech Documentation for QRectF in QT assistant 4.x. As soon as Qt3 compatibility is dropped this typedef will disappear.

Definition at line 45 of file qwt_double_rect.h.

7.1.2.3 QSizeF QwtDoubleSize

This is a typedef, see Trolltech Documentation for QSizeF in QT assistant 4.x. As soon as Qt3 compatibility is dropped this typedef will disappear.

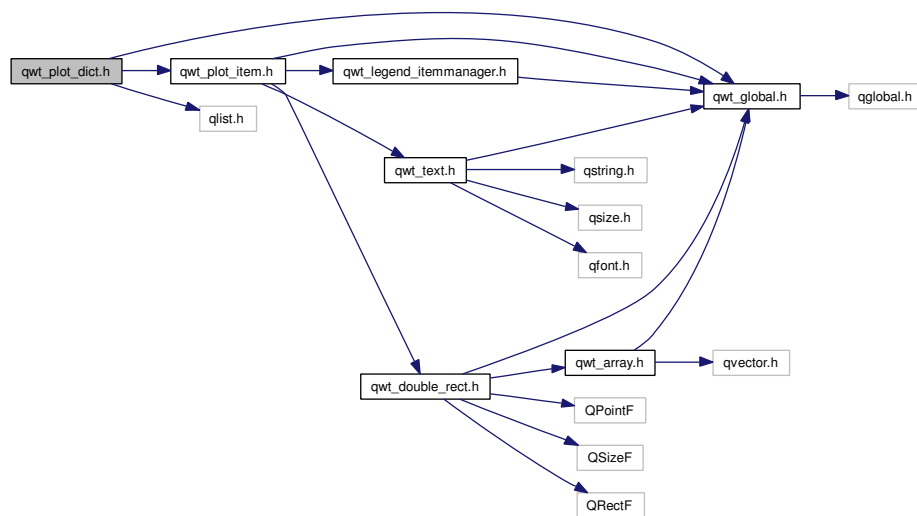
Definition at line 37 of file qwt_double_rect.h.

7.2 qwt_plot_dict.h File Reference

7.2.1 Detailed Description

Definition in file [qwt_plot_dict.h](#).

Include dependency graph for qwt_plot_dict.h:



Classes

- class [QwtPlotDict](#)
A dictionary for plot items.

Typedefs

- typedef QList< [QwtPlotItem](#) * >::ConstIterator [QwtPlotItemIterator](#)
- typedef QList< [QwtPlotItem](#) * > [QwtPlotItemList](#)

7.2.2 Typedef Documentation

7.2.2.1 typedef QList< [QwtPlotItem](#) * > [QwtPlotItemList](#)

See QT 4.x assistant documentation for QList.

Definition at line 30 of file qwt_plot_dict.h.

8 Qwt User's Guide Page Documentation

8.1 Qwt License, Version 1.0

Qwt License
Version 1.0, January 1, 2003

The Qwt library and included programs are provided under the terms of the GNU LESSER GENERAL PUBLIC LICENSE (LGPL) with the following exceptions:

1. Widgets that are subclassed from Qwt widgets do not constitute a derivative work.
2. Static linking of applications and widgets to the Qwt library does not constitute a derivative work and does not require the author to provide source code for the application or widget, use the shared Qwt libraries, or link their applications or widgets against a user-supplied version of Qwt.

If you link the application or widget to a modified version of Qwt, then the changes to Qwt must be provided under the terms of the LGPL in sections 1, 2, and 4.

3. You do not have to provide a copy of the Qwt license with programs that are linked to the Qwt library, nor do you have to identify the Qwt license in your program or documentation as required by section 6 of the LGPL.

However, programs must still identify their use of Qwt. The following example statement can be included in user documentation to satisfy this requirement:

[program/widget] is based in part on the work of the Qwt project (<http://qwt.sf.net>).

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does

and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2,

instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the

copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version,

but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
```

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

8.2 INSTALL

Introduction =====

Qwt uses qmake to build all its components and examples.
qmake is part of a Qt distribution.

qmake reads project files, that contain the options and rules how to build a certain project. A project file ends with the suffix "*.pro". Files that end with the suffix "*.pri" are included by the project files and contain definitions, that are common for several project files.

qwtconfig.pri is read by all project files of the Qwt package.
So the first step is to edit qwtconfig.pri to adjust it to your needs.

MathML Extension =====

Qwt/Qt4 supports the MathML render engine from the Qt solutions package, that is only available with a commercial Qt license.

You need a release of qtmmlwidget >= 2.1.
Copy the files qtmmlwidget.[cpp|h] to textengines/mathml.

A) Unix Qt3/Qt4 =====

```
qmake
make
make install
```

If you have installed a shared library it's path has to be known to the run-time linker of your operating system. On Linux systems read "man ldconfig" (or google for it). Another option is to use the LD_LIBRARY_PATH (on some systems LIBPATH is used instead, on MacOSX it is called DYLD_LIBRARY_PATH) environment variable.

If you only want to check the Qwt examples without installing something, you can set the LD_LIBRARY_PATH to the lib directory of your local build.

If you didn't enable autobuilding of the examples in `qwtconfig.pri` you have to build the examples this way:

```
cd examples
qmake
make
```

B) Win32/MSVC Qt3/Qt4
=====

Please read the `qmake` documentation how to convert your `*.pro` files into your development environment.

F.e MSVC with `nmake`:
`qmake qwt.pro`
`nmake`

If you didn't enable autobuilding of the examples in `qwtconfig.pri` you have to build the examples this way:

```
cd examples
qmake examples.pro
nmake
```

`admin/msvc-qmake.bat` helps users of Visual Studio users to generate makefiles or project files (`.dsp` for MSVC-6.0 or `vcproj` for MSVC.NET) for Qwt.

To generate makefiles, type: `"admin\msvc-qmake"`
To generate project files, type: `"admin\msvc-qmake vc"`

When you have built a Qwt DLL you need to add the following define to your compiler flags: `QWT_DLL`.

Windows doesn't like mixing of debug and release binaries. Most of the problems with using the Qwt designer plugin are because of trying to load a Qwt debug library into a designer release executable.

C) Win32/MinGW Qt4
=====

C1) Windows Shell

Start a Windows Shell, where Qt4 is initialized. (F.e. with `"Programs->Qt by Trolltech ...->Qt 4.x.x Command Prompt"`).

```
qmake qwt.pro
make
```

If you didn't enable autobuilding of the examples in `qwtconfig.pri` you have to build the examples this way:

```
cd examples
qmake examples.pro
make
make install
```

C2) MSYS Shell Qt >= 4.3.0

Support for the MSYS Shell has been improved in Qt 4.3.0. Now building Qwt from the MSYS Shell works exactly like in UNIX or in the Windows Shell - or at least it should: because of a bug in Qt 4.3.0 you always have to do a `"qmake -r"`.

C3) MSYS Shell Qt < 4.3.0

For Qt < 4.3.0 you have to set the MINGW_IN_SHELL variable.
make will run into errors with the subdirs target, that can be
ignored (make -i).

```
export MINGW_IN_SHELL=1;
```

```
qmake
make -i
make -i install
```

If you didn't enable autobuilding of the examples in qwtconfig.pri
you have to build the examples this way:

```
cd examples
qmake examples.pro
make -i
make -i install
```

C1-C3)

When you have built a Qwt DLL you need to add QWT_DLL to your compiler
flags. If you are using qmake for your own builds this done by adding
the following line to your profile: "DEFINES += QWT_DLL".

Windows doesn't like mixing of debug and release binaries. Most
of the problems with using the Qwt designer plugin are because
of trying to load a Qwt debug library into a designer release
executable.

D) MacOSX

Well, the Mac is only another Unix system. So read the instructions in A).

In the recent Qt4 releases the default target of qmake is to generate
XCode project files instead of makefiles. So you might need to do the
following:

```
qmake -spec macx-g++
...
```

D) Qtopia Core

I only tested Qwt with Qtopia Core in qvfb (Virtual Framebuffer Devivce)
Emulator on my Linux box. To build Qwt for the emulator was as simple as
for a regular Unix build.

```
qmake
make
```

E) Qtopia (!= Qtopia Core)

I once compiled the Qwt library against Qtopia 4.2.0 successfully - but
not more. It should be possible to build and install Qwt, but it's
not done yet.

Good luck !

8.3 Curve Plots

8.4 Scatter Plot

8.5 Spectrogram, Contour Plot

/*!

8.6 Histogram

8.7 Dials, Compasses, Knobs, Wheels, Sliders, Thermos

8.8 Deprecated List

Class [QwtPlotPrintFilter](#) In Qwt 5.0 the design of [QwtPlot](#) allows/recommends writing individual Qwt-PlotItems, that are not known to [QwtPlotPrintFilter](#). So this concept is outdated and [QwtPlotPrintFilter](#) will be removed/replaced in Qwt 6.x.

Class [QwtRect](#) Use [QwtClipper](#) instead.

8.9 Todo List

Class [QwtThermo](#) Improve the support for a logarithmic range and/or scale.

Index

- ~QwtAbstractScale
 - QwtAbstractScale, [15](#)
- ~QwtAbstractScaleDraw
 - QwtAbstractScaleDraw, [21](#)
- ~QwtAbstractSlider
 - QwtAbstractSlider, [30](#)
- ~QwtAlphaColorMap
 - QwtAlphaColorMap, [38](#)
- ~QwtAnalogClock
 - QwtAnalogClock, [42](#)
- ~QwtArrowButton
 - QwtArrowButton, [48](#)
- ~QwtColorMap
 - QwtColorMap, [52](#)
- ~QwtCompass
 - QwtCompass, [55](#)
- ~QwtCounter
 - QwtCounter, [68](#)
- ~QwtCurveFitter
 - QwtCurveFitter, [76](#)
- ~QwtData
 - QwtData, [77](#)
- ~QwtDial
 - QwtDial, [82](#)
- ~QwtDialNeedle
 - QwtDialNeedle, [95](#)
- ~QwtDoubleRange
 - QwtDoubleRange, [109](#)
- ~QwtDynGridLayout
 - QwtDynGridLayout, [116](#)
- ~QwtEventPattern
 - QwtEventPattern, [124](#)
- ~QwtKnob
 - QwtKnob, [130](#)
- ~QwtLegend
 - QwtLegend, [136](#)
- ~QwtLegendItem
 - QwtLegendItem, [141](#)
- ~QwtLinearColorMap
 - QwtLinearColorMap, [150](#)
- ~QwtMagnifier
 - QwtMagnifier, [158](#)
- ~QwtMathMLTextEngine
 - QwtMathMLTextEngine, [165](#)
- ~QwtPanner
 - QwtPanner, [175](#)
- ~QwtPicker
 - QwtPicker, [186](#)
- ~QwtPickerMachine
 - QwtPickerMachine, [204](#)
- ~QwtPlainTextEngine
 - QwtPlainTextEngine, [206](#)
- ~QwtPlot
 - QwtPlot, [212](#)
- ~QwtPlotCanvas
 - QwtPlotCanvas, [235](#)
- ~QwtPlotCurve
 - QwtPlotCurve, [242](#)
- ~QwtPlotDict
 - QwtPlotDict, [256](#)
- ~QwtPlotGrid
 - QwtPlotGrid, [259](#)
- ~QwtPlotItem
 - QwtPlotItem, [266](#)
- ~QwtPlotLayout
 - QwtPlotLayout, [276](#)
- ~QwtPlotMagnifier
 - QwtPlotMagnifier, [284](#)
- ~QwtPlotMarker
 - QwtPlotMarker, [288](#)
- ~QwtPlotPanner
 - QwtPlotPanner, [293](#)
- ~QwtPlotPrintFilter
 - QwtPlotPrintFilter, [304](#)
- ~QwtPlotRasterItem
 - QwtPlotRasterItem, [308](#)
- ~QwtPlotScaleItem
 - QwtPlotScaleItem, [312](#)
- ~QwtPlotSpectrogram
 - QwtPlotSpectrogram, [320](#)
- ~QwtPlotSvgItem
 - QwtPlotSvgItem, [328](#)
- ~QwtRoundScaleDraw
 - QwtRoundScaleDraw, [349](#)
- ~QwtScaleDraw
 - QwtScaleDraw, [360](#)
- ~QwtScaleEngine
 - QwtScaleEngine, [370](#)
- ~QwtScaleMap
 - QwtScaleMap, [375](#)
- ~QwtScaleWidget
 - QwtScaleWidget, [381](#)
- ~QwtSpline
 - QwtSpline, [403](#)
- ~QwtSymbol
 - QwtSymbol, [407](#)
- ~QwtText
 - QwtText, [414](#)
- ~QwtTextEngine
 - QwtTextEngine, [422](#)

- ~QwtTextLabel
 - QwtTextLabel, [425](#)
- ~QwtThermo
 - QwtThermo, [431](#)
- ~QwtWheel
 - QwtWheel, [439](#)
- abstractScaleDraw
 - QwtAbstractScale, [19](#)
- accept
 - QwtPicker, [194](#)
 - QwtPlotZoomer, [339](#)
- activate
 - QwtPlotLayout, [280](#)
- addColorStop
 - QwtLinearColorMap, [151](#)
- addItem
 - QwtDynGridLayout, [117](#)
- alarmBrush
 - QwtThermo, [433](#)
- alarmColor
 - QwtThermo, [433](#)
- alarmEnabled
 - QwtThermo, [434](#)
- alarmLevel
 - QwtThermo, [434](#)
- align
 - QwtLinearScaleEngine, [154](#)
- alignCanvasToScales
 - QwtPlotLayout, [278](#)
- alignLegend
 - QwtPlotLayout, [282](#)
- Alignment
 - QwtScaleDraw, [359](#)
- alignment
 - QwtScaleDraw, [362](#)
 - QwtScaleWidget, [388](#)
- alignScales
 - QwtPlotLayout, [283](#)
- alpha
 - QwtPlotRasterItem, [309](#)
- append
 - QwtPicker, [195](#)
 - QwtPlotPicker, [302](#)
- appended
 - QwtPicker, [193](#)
 - QwtPlotPicker, [299](#)
- apply
 - QwtPlotPrintFilter, [306](#)
- arrowSize
 - QwtArrowButton, [50](#)
- arrowType
 - QwtArrowButton, [48](#)
- attach
 - QwtPlotItem, [266](#)
- Attribute
 - QwtScaleEngine, [369](#)
- attributes
 - QwtScaleEngine, [371](#)
- autoDelete
 - QwtPlotDict, [256](#)
- autoRefresh
 - QwtPlot, [229](#)
- autoReplot
 - QwtPlot, [213](#)
- autoScale
 - QwtAbstractScale, [17](#)
 - QwtLinearScaleEngine, [153](#)
 - QwtLog10ScaleEngine, [156](#)
 - QwtScaleEngine, [372](#)
- Axis
 - QwtPlot, [212](#)
- axisAutoScale
 - QwtPlot, [219](#)
- axisEnabled
 - QwtPlot, [219](#)
- axisFont
 - QwtPlot, [220](#)
- axisMaxMajor
 - QwtPlot, [225](#)
- axisMaxMinor
 - QwtPlot, [226](#)
- axisScaleDiv
 - QwtPlot, [222](#)
- axisScaleDraw
 - QwtPlot, [223](#)
- axisScaleEngine
 - QwtPlot, [218](#)
- axisStepSize
 - QwtPlot, [221](#)
- axisTitle
 - QwtPlot, [225](#)
- axisValid
 - QwtPlot, [230](#)
- axisWidget
 - QwtPlot, [223](#), [224](#)
- backgroundBrush
 - QwtText, [418](#)
- backgroundPen
 - QwtText, [417](#)
- baseline
 - QwtPlotCurve, [249](#)
- begin
 - QwtPicker, [194](#)
 - QwtPlotZoomer, [339](#)
- BGSTYLE
 - QwtSlider, [395](#)

- bgStyle
 - QwtSlider, 396
- borderDistance
 - QwtPlotScaleItem, 316
- borderWidth
 - QwtKnob, 131
 - QwtSlider, 397
 - QwtThermo, 432
- boundingLabelRect
 - QwtScaleDraw, 366
- boundingRect
 - QwtArrayData, 47
 - QwtCPointerData, 75
 - QwtData, 78
 - QwtDial, 87
 - QwtPlotCurve, 246
 - QwtPlotItem, 272
 - QwtPlotMarker, 292
 - QwtPlotSpectrogram, 322
 - QwtPlotSvgItem, 329
- brush
 - QwtPlotCurve, 248
 - QwtSymbol, 409
- buildInterval
 - QwtScaleEngine, 374
- buildNaturalSpline
 - QwtSpline, 404
- buildPeriodicSpline
 - QwtSpline, 404
- Button
 - QwtCounter, 68
- buttonReleased
 - QwtCounter, 72
- CachePolicy
 - QwtPlotRasterItem, 308
- cachePolicy
 - QwtPlotRasterItem, 309
- canvas
 - QwtPlot, 216
 - QwtPlotMagnifier, 285
 - QwtPlotPanner, 294
 - QwtPlotPicker, 298
- canvasBackground
 - QwtPlot, 216
- canvasLineWidth
 - QwtPlot, 217
- canvasMap
 - QwtPlot, 217
- canvasMargin
 - QwtPlotLayout, 277
- canvasRect
 - QwtPlotLayout, 282
- ceil125
 - QwtScaleArithmetic, 353
- ceilEps
 - QwtScaleArithmetic, 353
- center
 - QwtRoundScaleDraw, 350
- changed
 - QwtPicker, 193
- checked
 - QwtLegendItem, 146
- clear
 - QwtLegend, 138
 - QwtPlot, 229
 - QwtTextLabel, 426
- clicked
 - QwtLegendItem, 146
- clip
 - QwtPainter, 174
 - QwtRect, 345
- closePolyline
 - QwtPlotCurve, 255
- color
 - QwtAlphaColorMap, 39
 - QwtColorMap, 53
 - QwtPlotPrintFilter, 305
 - QwtText, 417
- color1
 - QwtLinearColorMap, 151
- color2
 - QwtLinearColorMap, 152
- colorIndex
 - QwtColorMap, 53
 - QwtLinearColorMap, 152
- colorMap
 - QwtPlotSpectrogram, 321
- colorStops
 - QwtLinearColorMap, 151
- colorTable
 - QwtColorMap, 53
- columnsForWidth
 - QwtDynGridLayout, 119
- Command
 - QwtPickerMachine, 204
- compareEps
 - QwtScaleArithmetic, 353
- contains
 - QwtDoubleInterval, 105
 - QwtScaleEngine, 373
- contentsRect
 - QwtDial, 88
- contentsWidget
 - QwtLegend, 137
- contourLevels
 - QwtPlotSpectrogram, 324
- contourLines

- QwtRasterData, 344
- contourPen
 - QwtPlotSpectrogram, 323
- contourRasterSize
 - QwtPlotSpectrogram, 326
- copy
 - QwtAlphaColorMap, 39
 - QwtArrayData, 46
 - QwtColorMap, 52
 - QwtCPointerData, 74
 - QwtData, 77
 - QwtLinearColorMap, 150
 - QwtPolygonFData, 341
 - QwtRasterData, 343
- cursor
 - QwtPanner, 177
- CurveAttribute
 - QwtPlotCurve, 241
- curvePen
 - QwtLegendItem, 144
- CurveStyle
 - QwtPlotCurve, 241
- curveType
 - QwtPlotCurve, 242
- data
 - QwtPlotCurve, 245
 - QwtPlotSpectrogram, 321
- dataSize
 - QwtPlotCurve, 245
- defaultContourPen
 - QwtPlotSpectrogram, 323
- detach
 - QwtPlotItem, 266
- detachItems
 - QwtPlotDict, 257
- deviceClipping
 - QwtPainter, 171
- dimForLength
 - QwtScaleWidget, 387
- discardRaster
 - QwtRasterData, 343
- DisplayMode
 - QwtPicker, 185
 - QwtPlotSpectrogram, 319
- displayPolicy
 - QwtLegend, 136
- divideInterval
 - QwtScaleEngine, 374
- divideScale
 - QwtLinearScaleEngine, 154
 - QwtLog10ScaleEngine, 156
 - QwtScaleEngine, 373
- draw
 - QwtAbstractScaleDraw, 24
 - QwtCompassMagnetNeedle, 60
 - QwtCompassRose, 62
 - QwtCompassWindArrow, 65
 - QwtDialNeedle, 95
 - QwtDialSimpleNeedle, 100
 - QwtKnob, 133
 - QwtMathMLTextEngine, 166
 - QwtPlainTextEngine, 207
 - QwtPlotCurve, 250, 251
 - QwtPlotGrid, 263
 - QwtPlotItem, 272
 - QwtPlotMarker, 291
 - QwtPlotRasterItem, 310
 - QwtPlotScaleItem, 317
 - QwtPlotSpectrogram, 325
 - QwtPlotSvgItem, 329
 - QwtRichTextEngine, 346
 - QwtScaleWidget, 389
 - QwtSimpleCompassRose, 392
 - QwtSlider, 399
 - QwtSymbol, 410
 - QwtText, 420
 - QwtTextEngine, 424
 - QwtThermo, 437
 - QwtWheel, 442
- drawArrow
 - QwtArrowButton, 49
- drawArrowNeedle
 - QwtDialSimpleNeedle, 100
- drawBackbone
 - QwtAbstractScaleDraw, 26
 - QwtRoundScaleDraw, 351
 - QwtScaleDraw, 367
- drawButtonLabel
 - QwtArrowButton, 49
- drawCanvas
 - QwtPlot, 230
 - QwtPlotCanvas, 237
- drawContents
 - QwtDial, 91
 - QwtPlotCanvas, 237
 - QwtTextLabel, 428
- drawContourLines
 - QwtPlotSpectrogram, 326
- drawCurve
 - QwtPlotCurve, 251
- drawDots
 - QwtPlotCurve, 253
- drawEllipse
 - QwtPainter, 173
- drawFocusIndicator
 - QwtDial, 91
 - QwtPlotCanvas, 237

- drawFrame
 - QwtDial, [91](#)
- drawHand
 - QwtAnalogClock, [44](#)
- drawIdentifier
 - QwtLegendItem, [145](#)
- drawItem
 - QwtLegendItem, [145](#)
- drawItems
 - QwtPlot, [230](#)
- drawKnob
 - QwtDialNeedle, [96](#)
 - QwtKnob, [133](#)
- drawLabel
 - QwtAbstractScaleDraw, [27](#)
 - QwtRoundScaleDraw, [352](#)
 - QwtScaleDraw, [368](#)
- drawLine
 - QwtPainter, [173](#)
- drawLines
 - QwtPlotCurve, [252](#)
- drawMarker
 - QwtKnob, [134](#)
- drawNeedle
 - QwtAnalogClock, [43](#)
 - QwtDial, [92](#)
- drawPie
 - QwtPainter, [173](#)
- drawPoint
 - QwtPainter, [174](#)
- drawPointer
 - QwtCompassMagnetNeedle, [61](#)
- drawPolygon
 - QwtPainter, [173](#)
- drawPolyline
 - QwtPainter, [173](#)
- drawRayNeedle
 - QwtDialSimpleNeedle, [101](#)
- drawRect
 - QwtPainter, [172](#)
- drawRose
 - QwtCompass, [57](#)
 - QwtSimpleCompassRose, [392](#)
- drawRoundFrame
 - QwtPainter, [174](#)
- drawRubberBand
 - QwtPicker, [192](#)
- drawScale
 - QwtDial, [92](#)
- drawScaleContents
 - QwtCompass, [57](#)
 - QwtDial, [92](#)
- drawSimpleRichText
 - QwtPainter, [172](#)
- drawSlider
 - QwtSlider, [400](#)
- drawSteps
 - QwtPlotCurve, [254](#)
- drawSticks
 - QwtPlotCurve, [253](#)
- drawStyle1Needle
 - QwtCompassWindArrow, [65](#)
- drawStyle2Needle
 - QwtCompassWindArrow, [65](#)
- drawSymbols
 - QwtPlotCurve, [252](#)
- drawText
 - QwtLegendItem, [147](#)
 - QwtPainter, [171](#), [172](#)
 - QwtTextLabel, [428](#)
- drawThermo
 - QwtThermo, [437](#)
- drawThinNeedle
 - QwtCompassMagnetNeedle, [61](#)
- drawThumb
 - QwtSlider, [400](#)
- drawTick
 - QwtAbstractScaleDraw, [26](#)
 - QwtRoundScaleDraw, [351](#)
 - QwtScaleDraw, [367](#)
- drawTitle
 - QwtScaleWidget, [388](#)
- drawTracker
 - QwtPicker, [192](#)
- drawTriangleNeedle
 - QwtCompassMagnetNeedle, [60](#)
- drawWheel
 - QwtWheel, [442](#)
- drawWheelBackground
 - QwtWheel, [443](#)
- editable
 - QwtCounter, [68](#)
- enableAxis
 - QwtPlot, [219](#)
- enableComponent
 - QwtAbstractScaleDraw, [23](#)
- enableX
 - QwtPlotGrid, [259](#)
- enableXMin
 - QwtPlotGrid, [260](#)
- enableY
 - QwtPlotGrid, [259](#)
- enableYMin
 - QwtPlotGrid, [260](#)
- end
 - QwtPicker, [195](#)
 - QwtPlotPicker, [302](#)

- QwtPlotZoomer, 339
- endBorderDist
 - QwtScaleWidget, 383
- event
 - QwtCounter, 72
 - QwtPlot, 228
- eventFilter
 - QwtMagnifier, 162
 - QwtPanner, 177
 - QwtPicker, 191
- exactPrevValue
 - QwtDoubleRange, 114
- exactValue
 - QwtDoubleRange, 113
- expandLineBreaks
 - QwtPlotLayout, 283
- extend
 - QwtDoubleInterval, 106
- extent
 - QwtAbstractScaleDraw, 25
 - QwtRoundScaleDraw, 350
 - QwtScaleDraw, 361
- fillBrush
 - QwtThermo, 433
- fillColor
 - QwtThermo, 433
- fillCurve
 - QwtPlotCurve, 254
- fillRect
 - QwtPainter, 172
- find
 - QwtLegend, 138
- fitValue
 - QwtAbstractSlider, 33
 - QwtDoubleRange, 113
- floor125
 - QwtScaleArithmetic, 354
- floorEps
 - QwtScaleArithmetic, 353
- FocusIndicator
 - QwtPlotCanvas, 234
- focusIndicator
 - QwtPlotCanvas, 236
- font
 - QwtPlotPrintFilter, 305
 - QwtPlotScaleItem, 314
 - QwtText, 415
- fontChange
 - QwtSlider, 401
 - QwtThermo, 437
- Format
 - QwtColorMap, 51
- format
 - QwtColorMap, 52
- frameShadow
 - QwtDial, 82
- getAbortKey
 - QwtPanner, 176
- getBorderDistHint
 - QwtScaleDraw, 360
 - QwtScaleWidget, 383
- getMinBorderDist
 - QwtScaleWidget, 383
- getMouseButton
 - QwtMagnifier, 160
 - QwtPanner, 176
- getScrollMode
 - QwtAbstractSlider, 37
 - QwtDial, 94
 - QwtSlider, 399
 - QwtWheel, 443
- getValue
 - QwtAbstractSlider, 36
 - QwtDial, 94
 - QwtSlider, 399
 - QwtWheel, 443
- getZoomInKey
 - QwtMagnifier, 162
- getZoomOutKey
 - QwtMagnifier, 162
- Hand
 - QwtAnalogClock, 41
- hand
 - QwtAnalogClock, 42
- hasComponent
 - QwtAbstractScaleDraw, 23
- hasHeightForWidth
 - QwtDynGridLayout, 118
- hasVisibleBackground
 - QwtDial, 82
- hBound
 - QwtScaleDiv, 357
- heightForWidth
 - QwtDynGridLayout, 118
 - QwtLegend, 139
 - QwtMathMLTextEngine, 166
 - QwtPlainTextEngine, 207
 - QwtRichTextEngine, 346
 - QwtText, 419
 - QwtTextEngine, 423
 - QwtTextLabel, 427
- hide
 - QwtPlotItem, 270
- hiMargin
 - QwtScaleEngine, 372

- IdentifierMode
 - QwtLegendItem, 141
- identifierMode
 - QwtLegend, 137
 - QwtLegendItem, 143
- identifierWidth
 - QwtLegendItem, 143
- incPages
 - QwtDoubleRange, 113
- incSteps
 - QwtCounter, 69
- incValue
 - QwtAbstractSlider, 33
 - QwtDoubleRange, 112
- indent
 - QwtTextLabel, 426
- init
 - QwtPlotCurve, 251
- initKeyPattern
 - QwtEventPattern, 124
- initMousePattern
 - QwtEventPattern, 124
- initRaster
 - QwtRasterData, 343
- insert
 - QwtLegend, 137
- insertLegend
 - QwtPlot, 226
- intersect
 - QwtDoubleInterval, 106
- intersects
 - QwtDoubleInterval, 105
- interval
 - QwtScaleDiv, 356
- invalidate
 - QwtDoubleInterval, 107
 - QwtPlotLayout, 281
 - QwtScaleDiv, 357
- invalidateCache
 - QwtAbstractScaleDraw, 27
 - QwtPlotRasterItem, 310
- invalidatePaintCache
 - QwtPlotCanvas, 237
- invert
 - QwtScaleDiv, 358
- inverted
 - QwtDoubleInterval, 103
- invTransform
 - QwtPlot, 217
 - QwtPlotItem, 275
 - QwtPlotPicker, 300
 - QwtScaleMap, 377
- invXForm
 - QwtScaleTransformation, 379
- isActive
 - QwtPicker, 191
- isAxisEnabled
 - QwtPlotMagnifier, 285
 - QwtPlotPanner, 294
- isChecked
 - QwtLegendItem, 145
- isDown
 - QwtLegendItem, 146
- isEmpty
 - QwtDynGridLayout, 118
 - QwtLegend, 138
 - QwtText, 415
- isEnabled
 - QwtMagnifier, 159
 - QwtPanner, 176
 - QwtPicker, 190
- isNull
 - QwtDoubleInterval, 107
 - QwtText, 415
- isOrientationEnabled
 - QwtPanner, 177
- isReadOnly
 - QwtAbstractSlider, 32
- isScaleDivFromAxis
 - QwtPlotScaleItem, 313
- isValid
 - QwtAbstractSlider, 32
 - QwtDoubleInterval, 107
 - QwtDoubleRange, 110
 - QwtScaleDiv, 358
 - QwtSpline, 404
- isVisible
 - QwtPlotItem, 270
- Item
 - QwtPlotPrintFilter, 304
- itemChanged
 - QwtPlotItem, 271
- itemCount
 - QwtDynGridLayout, 119
 - QwtLegend, 138
- itemList
 - QwtPlotDict, 257
- itemMode
 - QwtLegendItem, 142
- keyFactor
 - QwtMagnifier, 161
- keyMatch
 - QwtEventPattern, 126, 127
- keyPattern
 - QwtEventPattern, 125, 126
- KeyPatternCode
 - QwtEventPattern, 123

- keyPressEvent
 - QwtAbstractSlider, 36
 - QwtArrowButton, 50
 - QwtCompass, 58
 - QwtCounter, 72
 - QwtDial, 90
 - QwtLegendItem, 147
- keyReleaseEvent
 - QwtLegendItem, 147
- knobWidth
 - QwtKnob, 131
- label
 - QwtAbstractScaleDraw, 25
 - QwtDialScaleDraw, 97
 - QwtPlotMarker, 291
- labelAlignment
 - QwtPlotMarker, 291
 - QwtScaleDraw, 364
- labelMap
 - QwtCompass, 56
- labelMatrix
 - QwtScaleDraw, 367
- labelPosition
 - QwtScaleDraw, 365
- labelRect
 - QwtArrowButton, 50
 - QwtScaleDraw, 366
- labelRotation
 - QwtScaleDraw, 365
- labelSize
 - QwtScaleDraw, 366
- LayoutAttribute
 - QwtText, 413
- layoutContents
 - QwtLegend, 139
- layoutGrid
 - QwtDynGridLayout, 119
- layoutItems
 - QwtDynGridLayout, 117
- layoutLegend
 - QwtPlotLayout, 282
- layoutScale
 - QwtScaleWidget, 389
- layoutSlider
 - QwtSlider, 401
- layoutThermo
 - QwtThermo, 437
- layoutWheel
 - QwtWheel, 442
- lBound
 - QwtScaleDiv, 356
- legend
 - QwtPlot, 227
 - legendChecked
 - QwtPlot, 228
 - legendClicked
 - QwtPlot, 228
 - LegendDisplayPolicy
 - QwtLegend, 135
 - legendItem
 - QwtPlotItem, 273
 - legendItemChecked
 - QwtPlot, 230
 - legendItemClicked
 - QwtPlot, 229
 - LegendItemMode
 - QwtLegend, 136
 - LegendPosition
 - QwtPlot, 212
 - legendPosition
 - QwtPlotLayout, 279
 - legendRatio
 - QwtPlotLayout, 280
 - legendRect
 - QwtPlotLayout, 281
 - length
 - QwtScaleDraw, 363
 - limited
 - QwtDoubleInterval, 103
 - linePen
 - QwtPlotMarker, 290
 - LineStyle
 - QwtPlotMarker, 287
 - lineStyle
 - QwtPlotMarker, 289
 - lineWidth
 - QwtDial, 83
 - loadData
 - QwtPlotSvgItem, 329
 - loadFile
 - QwtPlotSvgItem, 329
 - log10
 - QwtLog10ScaleEngine, 157
 - loMargin
 - QwtScaleEngine, 372
 - majPen
 - QwtPlotGrid, 262
 - majTickLength
 - QwtAbstractScaleDraw, 24
 - map
 - QwtAbstractScaleDraw, 22
 - margin
 - QwtPlot, 214
 - QwtPlotLayout, 277
 - QwtScaleWidget, 384
 - QwtTextLabel, 427

- mass
 - QwtAbstractSlider, 31
 - QwtWheel, 440
- maxCols
 - QwtDynGridLayout, 116
- maxItemWidth
 - QwtDynGridLayout, 117
- maxLabelHeight
 - QwtScaleDraw, 365
- maxLabelWidth
 - QwtScaleDraw, 365
- maxScaleArc
 - QwtDial, 86
- maxStackDepth
 - QwtPlotZoomer, 335
- maxVal
 - QwtCounter, 71
- maxValue
 - QwtDoubleInterval, 104
 - QwtDoubleRange, 112
 - QwtThermo, 434
- maxXValue
 - QwtPlotCurve, 246
- maxYValue
 - QwtPlotCurve, 246
- metricsMap
 - QwtPainter, 171
- mightRender
 - QwtMathMLTextEngine, 167
 - QwtPlainTextEngine, 207
 - QwtRichTextEngine, 347
 - QwtTextEngine, 423
- minimumExtent
 - QwtAbstractScaleDraw, 26
- minimumSizeHint
 - QwtArrowButton, 49
 - QwtDial, 88
 - QwtKnob, 132
 - QwtPlot, 228
 - QwtPlotLayout, 280
 - QwtScaleWidget, 387
 - QwtSlider, 398
 - QwtTextLabel, 427
 - QwtThermo, 436
 - QwtWheel, 442
- minLabelDist
 - QwtScaleDraw, 360
- minLength
 - QwtScaleDraw, 361
- minPen
 - QwtPlotGrid, 263
- minScaleArc
 - QwtDial, 86
- minVal
 - QwtCounter, 70
- minValue
 - QwtDoubleInterval, 104
 - QwtDoubleRange, 112
 - QwtThermo, 435
- minXValue
 - QwtPlotCurve, 246
- minYValue
 - QwtPlotCurve, 246
- minZoomSize
 - QwtPlotZoomer, 338
- Mode
 - QwtDial, 81
 - QwtLinearColorMap, 149
- mode
 - QwtDial, 84
 - QwtLinearColorMap, 150
- mouseFactor
 - QwtMagnifier, 159
- mouseMatch
 - QwtEventPattern, 126, 127
- mouseMoveEvent
 - QwtAbstractSlider, 35
- mousePattern
 - QwtEventPattern, 125
- MousePatternCode
 - QwtEventPattern, 122
- mousePressEvent
 - QwtAbstractSlider, 35
 - QwtLegendItem, 147
- mouseReleaseEvent
 - QwtAbstractSlider, 35
 - QwtLegendItem, 147
- move
 - QwtPicker, 195
 - QwtPlotPicker, 302
 - QwtPlotZoomer, 336
 - QwtScaleDraw, 362
- moveBy
 - QwtPlotZoomer, 336
- moveCanvas
 - QwtPlotPanner, 295
- moveCenter
 - QwtRoundScaleDraw, 350
- moved
 - QwtPanner, 178
 - QwtPicker, 193
 - QwtPlotPicker, 299
- needle
 - QwtDial, 87
- normalized
 - QwtDoubleInterval, 103
- num

- QwtArrowButton, 48
- numButtons
 - QwtCounter, 69
- numCols
 - QwtDynGridLayout, 117
- numRows
 - QwtDynGridLayout, 116
- numThornLevels
 - QwtSimpleCompassRose, 391
- numThorns
 - QwtSimpleCompassRose, 391
- operator &
 - QwtDoubleInterval, 106
- operator!=
 - QwtDoubleInterval, 104
 - QwtScaleDiv, 356
 - QwtSymbol, 407
- operator=
 - QwtAbstractScaleDraw, 22
 - QwtAlphaColorMap, 38
 - QwtArrayData, 46
 - QwtCPointerData, 74
 - QwtData, 78
 - QwtLinearColorMap, 150
 - QwtPolygonFData, 341
 - QwtRoundScaleDraw, 349
 - QwtScaleDraw, 360
 - QwtText, 414
- operator==
 - QwtDoubleInterval, 104
 - QwtScaleDiv, 356
 - QwtSymbol, 407
- operator |
 - QwtDoubleInterval, 106, 107
- Options
 - QwtPlotPrintFilter, 304
- options
 - QwtPlotPrintFilter, 305
- orientation
 - QwtAbstractSlider, 31
 - QwtScaleDraw, 363
- orientations
 - QwtPanner, 177
- origin
 - QwtDial, 86
- p1
 - QwtScaleMap, 377
- p2
 - QwtScaleMap, 378
- pageSize
 - QwtDoubleRange, 112
- PaintAttribute
 - QwtPlotCanvas, 234
 - QwtPlotCurve, 241
 - QwtText, 413
- paintCache
 - QwtPlotCanvas, 236, 237
- paintEvent
 - QwtArrowButton, 49
 - QwtDial, 89
 - QwtKnob, 133
 - QwtLegendItem, 146
 - QwtPanner, 180
 - QwtPlotCanvas, 237
 - QwtScaleWidget, 389
 - QwtSlider, 400
 - QwtTextLabel, 428
 - QwtThermo, 437
 - QwtWheel, 442
- paintRect
 - QwtPlotItem, 274
- palette
 - QwtDialNeedle, 96
 - QwtPlotScaleItem, 314
- paletteChange
 - QwtWheel, 443
- panned
 - QwtPanner, 178
- parentWidget
 - QwtPicker, 191
- pen
 - QwtPlotCurve, 247
 - QwtSymbol, 409
- penWidth
 - QwtDialScaleDraw, 98
 - QwtScaleWidget, 385
- periodic
 - QwtDoubleRange, 111
- pickRect
 - QwtPicker, 192
- pipeWidth
 - QwtThermo, 434
- plot
 - QwtPlotCanvas, 235
 - QwtPlotItem, 266
 - QwtPlotMagnifier, 285
 - QwtPlotPanner, 294
 - QwtPlotPicker, 298
- plotLayout
 - QwtPlot, 214
- points
 - QwtSpline, 404
- polish
 - QwtCounter, 70
 - QwtPlot, 227
- pos

- QwtScaleDraw, 363
- position
 - QwtPlotScaleItem, 315
- pow10
 - QwtLog10ScaleEngine, 157
- pressed
 - QwtLegendItem, 146
- prevValue
 - QwtDoubleRange, 114
- print
 - QwtPlot, 213
- printCanvas
 - QwtPlot, 232
- printLegend
 - QwtPlot, 233
- printLegendItem
 - QwtPlot, 231
- printScale
 - QwtPlot, 232
- printTitle
 - QwtPlot, 231
- qwt_double_rect.h, 444
 - QwtDoublePoint, 444
 - QwtDoubleRect, 444
 - QwtDoubleSize, 445
- qwt_plot_dict.h, 445
 - QwtPlotItemList, 446
- QwtAbstractScale, 14
 - QwtAbstractScale, 15
- QwtAbstractScale
 - ~QwtAbstractScale, 15
 - abstractScaleDraw, 19
 - autoScale, 17
 - QwtAbstractScale, 15
 - rescale, 18
 - scaleChange, 19
 - scaleEngine, 18
 - scaleMap, 18
 - scaleMaxMajor, 17
 - scaleMaxMinor, 17
 - setAbstractScaleDraw, 19
 - setAutoScale, 16
 - setScale, 15, 16
 - setScaleEngine, 18
 - setScaleMaxMajor, 17
 - setScaleMaxMinor, 17
- QwtAbstractScaleDraw, 20
 - QwtAbstractScaleDraw, 21
- QwtAbstractScaleDraw
 - ~QwtAbstractScaleDraw, 21
 - draw, 24
 - drawBackbone, 26
 - drawLabel, 27
 - drawTick, 26
 - enableComponent, 23
 - extent, 25
 - hasComponent, 23
 - invalidateCache, 27
 - label, 25
 - majTickLength, 24
 - map, 22
 - minimumExtent, 26
 - operator=, 22
 - QwtAbstractScaleDraw, 21
 - ScaleComponent, 21
 - scaleDiv, 22
 - scaleMap, 26
 - setMinimumExtent, 25
 - setScaleDiv, 22
 - setSpacing, 24
 - setTickLength, 23
 - setTransformation, 22
 - spacing, 24
 - tickLabel, 27
 - tickLength, 24
- QwtAbstractSlider, 28
 - QwtAbstractSlider, 30
- QwtAbstractSlider
 - ~QwtAbstractSlider, 30
 - fitValue, 33
 - getScrollMode, 37
 - getValue, 36
 - incValue, 33
 - isReadOnly, 32
 - isValid, 32
 - keyPressEvent, 36
 - mass, 31
 - mouseMoveEvent, 35
 - mousePressEvent, 35
 - mouseReleaseEvent, 35
 - orientation, 31
 - QwtAbstractSlider, 30
 - ScrollMode, 29
 - setMass, 31
 - setOrientation, 31
 - setPosition, 34
 - setReadOnly, 34
 - setTracking, 30
 - setUpdateTime, 30
 - setValid, 32
 - setValue, 33
 - sliderMoved, 34
 - sliderPressed, 34
 - sliderReleased, 34
 - stopMoving, 30
 - timerEvent, 35
 - valueChange, 35

- valueChanged, 34
- wheelEvent, 36
- QwtAlphaColorMap, 37
 - QwtAlphaColorMap, 38
- QwtAlphaColorMap
 - ~QwtAlphaColorMap, 38
 - color, 39
 - copy, 39
 - operator=, 38
 - QwtAlphaColorMap, 38
 - rgb, 39
 - setColor, 39
- QwtAnalogClock, 40
 - QwtAnalogClock, 42
- QwtAnalogClock
 - ~QwtAnalogClock, 42
 - drawHand, 44
 - drawNeedle, 43
 - Hand, 41
 - hand, 42
 - QwtAnalogClock, 42
 - scaleLabel, 43
 - setCurrentTime, 43
 - setHand, 42
 - setTime, 43
- QwtArrayData, 44
 - QwtArrayData, 45
- QwtArrayData
 - boundingRect, 47
 - copy, 46
 - operator=, 46
 - QwtArrayData, 45
 - size, 46
 - x, 46
 - xData, 47
 - y, 46
 - yData, 47
- QwtArrowButton, 47
 - QwtArrowButton, 48
- QwtArrowButton
 - ~QwtArrowButton, 48
 - arrowSize, 50
 - arrowType, 48
 - drawArrow, 49
 - drawButtonLabel, 49
 - keyPressEvent, 50
 - labelRect, 50
 - minimumSizeHint, 49
 - num, 48
 - paintEvent, 49
 - QwtArrowButton, 48
 - sizeHint, 48
- QwtClipper, 50
- QwtColorMap, 51
 - QwtColorMap, 52
- QwtColorMap
 - ~QwtColorMap, 52
 - color, 53
 - colorIndex, 53
 - colorTable, 53
 - copy, 52
 - Format, 51
 - format, 52
 - QwtColorMap, 52
 - rgb, 52
- QwtCompass, 54
 - QwtCompass, 55
- QwtCompass
 - ~QwtCompass, 55
 - drawRose, 57
 - drawScaleContents, 57
 - keyPressEvent, 58
 - labelMap, 56
 - QwtCompass, 55
 - rose, 55, 56
 - scaleLabel, 57
 - setLabelMap, 56
 - setRose, 55
- QwtCompassMagnetNeedle, 58
 - QwtCompassMagnetNeedle, 60
- QwtCompassMagnetNeedle
 - draw, 60
 - drawPointer, 61
 - drawThinNeedle, 61
 - drawTriangleNeedle, 60
 - QwtCompassMagnetNeedle, 60
 - Style, 60
- QwtCompassRose, 62
- QwtCompassRose
 - draw, 62
- QwtCompassWindArrow, 63
 - QwtCompassWindArrow, 64
- QwtCompassWindArrow
 - draw, 65
 - drawStyle1Needle, 65
 - drawStyle2Needle, 65
 - QwtCompassWindArrow, 64
 - Style, 64
- QwtCounter, 66
 - QwtCounter, 68
- QwtCounter
 - ~QwtCounter, 68
 - Button, 68
 - buttonReleased, 72
 - editable, 68
 - event, 72
 - incSteps, 69
 - keyPressEvent, 72

- maxVal, 71
- minVal, 70
- numButtons, 69
- polish, 70
- QwtCounter, 68
- rangeChange, 73
- setEditable, 68
- setIncSteps, 69
- setMaxValue, 71
- setMinValue, 70
- setNumButtons, 68
- setStep, 70
- setStepButton1, 71
- setStepButton2, 71
- setStepButton3, 71
- setValue, 69
- sizeHint, 70
- step, 70
- stepButton1, 71
- stepButton2, 71
- stepButton3, 71
- value, 72
- valueChanged, 72
- QwtCPointerData, 73
 - QwtCPointerData, 74
- QwtCPointerData
 - boundingRect, 75
 - copy, 74
 - operator=, 74
 - QwtCPointerData, 74
 - size, 74
 - x, 74
 - xData, 75
 - y, 75
 - yData, 75
- QwtCurveFitter, 76
 - QwtCurveFitter, 76
- QwtCurveFitter
 - ~QwtCurveFitter, 76
 - QwtCurveFitter, 76
- QwtData, 76
 - QwtData, 77
- QwtData
 - ~QwtData, 77
 - boundingRect, 78
 - copy, 77
 - operator=, 78
 - QwtData, 77
 - size, 77
 - x, 78
 - y, 78
- QwtDial, 79
 - QwtDial, 82
- QwtDial
 - ~QwtDial, 82
 - boundingRect, 87
 - contentsRect, 88
 - drawContents, 91
 - drawFocusIndicator, 91
 - drawFrame, 91
 - drawNeedle, 92
 - drawScale, 92
 - drawScaleContents, 92
 - frameShadow, 82
 - getScrollMode, 94
 - getValue, 94
 - hasVisibleBackground, 82
 - keyPressEvent, 90
 - lineWidth, 83
 - maxScaleArc, 86
 - minimumSizeHint, 88
 - minScaleArc, 86
 - Mode, 81
 - mode, 84
 - needle, 87
 - origin, 86
 - paintEvent, 89
 - QwtDial, 82
 - rangeChange, 93
 - resizeEvent, 89
 - scaleContentsRect, 88
 - scaleDraw, 89
 - scaleLabel, 93
 - ScaleOptions, 81
 - setFrameShadow, 82
 - setLineWidth, 83
 - setMode, 83
 - setNeedle, 87
 - setOrigin, 86
 - setScale, 85
 - setScaleArc, 85
 - setScaleDraw, 89
 - setScaleOptions, 85
 - setScaleTicks, 86
 - setWrapping, 84
 - Shadow, 81
 - showBackground, 82
 - sizeHint, 88
 - updateMask, 90
 - updateScale, 93
 - valueChange, 93
 - wrapping, 84
- QwtDialNeedle, 94
 - QwtDialNeedle, 95
- QwtDialNeedle
 - ~QwtDialNeedle, 95
 - draw, 95
 - drawKnob, 96

- palette, 96
- QwtDialNeedle, 95
- setPalette, 95
- QwtDialScaleDraw, 96
 - QwtDialScaleDraw, 97
- QwtDialScaleDraw
 - label, 97
 - penWidth, 98
 - QwtDialScaleDraw, 97
 - setPenWidth, 98
- QwtDialSimpleNeedle, 98
 - QwtDialSimpleNeedle, 100
- QwtDialSimpleNeedle
 - draw, 100
 - drawArrowNeedle, 100
 - drawRayNeedle, 101
 - QwtDialSimpleNeedle, 100
 - setWidth, 101
 - Style, 100
 - width, 101
- QwtDoubleInterval, 101
 - QwtDoubleInterval, 102
- QwtDoubleInterval
 - contains, 105
 - extend, 106
 - intersect, 106
 - intersects, 105
 - invalidate, 107
 - inverted, 103
 - isNull, 107
 - isValid, 107
 - limited, 103
 - maxValue, 104
 - minValue, 104
 - normalized, 103
 - operator &, 106
 - operator!=, 104
 - operator==, 104
 - operator |, 106, 107
 - QwtDoubleInterval, 102
 - setInterval, 103
 - setMaxValue, 105
 - setMinValue, 105
 - symmetrize, 107
 - unite, 106
 - width, 104
- QwtDoublePoint
 - qwt_double_rect.h, 444
- QwtDoubleRange, 108
 - QwtDoubleRange, 109
- QwtDoubleRange
 - ~QwtDoubleRange, 109
 - exactPrevValue, 114
 - exactValue, 113
 - fitValue, 113
 - incPages, 113
 - incValue, 112
 - isValid, 110
 - maxValue, 112
 - minValue, 112
 - pageSize, 112
 - periodic, 111
 - prevValue, 114
 - QwtDoubleRange, 109
 - rangeChange, 114
 - setPeriodic, 111
 - setRange, 109
 - setStep, 111
 - setValid, 110
 - setValue, 110
 - step, 111
 - stepChange, 114
 - value, 110
 - valueChange, 114
- QwtDoubleRect
 - qwt_double_rect.h, 444
- QwtDoubleSize
 - qwt_double_rect.h, 445
- QwtDynGridLayout, 115
 - QwtDynGridLayout, 115, 116
- QwtDynGridLayout
 - ~QwtDynGridLayout, 116
 - addItem, 117
 - columnsForWidth, 119
 - hasHeightForWidth, 118
 - heightForWidth, 118
 - isEmpty, 118
 - itemCount, 119
 - layoutGrid, 119
 - layoutItems, 117
 - maxCols, 116
 - maxItemWidth, 117
 - numCols, 117
 - numRows, 116
 - QwtDynGridLayout, 115, 116
 - setGeometry, 118
 - setMaxCols, 116
 - sizeHint, 118
 - stretchGrid, 119
- QwtEventPattern, 120
 - QwtEventPattern, 124
- QwtEventPattern
 - ~QwtEventPattern, 124
 - initKeyPattern, 124
 - initMousePattern, 124
 - keyMatch, 126, 127
 - keyPattern, 125, 126
 - KeyPatternCode, 123

- mouseMatch, 126, 127
- mousePattern, 125
- MousePatternCode, 122
- QwtEventPattern, 124
- setKeyPattern, 125
- setMousePattern, 124, 125
- QwtEventPattern::KeyPattern, 128
- QwtEventPattern::MousePattern, 128
- QwtIntervalData, 128
- QwtKnob, 129
 - QwtKnob, 130
- QwtKnob
 - ~QwtKnob, 130
 - borderWidth, 131
 - draw, 133
 - drawKnob, 133
 - drawMarker, 134
 - knobWidth, 131
 - minimumSizeHint, 132
 - paintEvent, 133
 - QwtKnob, 130
 - resizeEvent, 133
 - scaleDraw, 132, 133
 - setBorderWidth, 131
 - setKnobWidth, 131
 - setScaleDraw, 132
 - setSymbol, 131
 - setTotalAngle, 131
 - sizeHint, 132
 - Symbol, 130
 - symbol, 132
 - totalAngle, 131
- QwtLegend, 134
 - QwtLegend, 136
- QwtLegend
 - ~QwtLegend, 136
 - clear, 138
 - contentsWidget, 137
 - displayPolicy, 136
 - find, 138
 - heightForWidth, 139
 - identifierMode, 137
 - insert, 137
 - isEmpty, 138
 - itemCount, 138
 - layoutContents, 139
 - LegendDisplayPolicy, 135
 - LegendItemMode, 136
 - QwtLegend, 136
 - remove, 137
 - resizeEvent, 139
 - setDisplayPolicy, 136
 - sizeHint, 138
- QwtLegendItem, 139
 - QwtLegendItem, 141
- QwtLegendItem
 - ~QwtLegendItem, 141
 - checked, 146
 - clicked, 146
 - curvePen, 144
 - drawIdentifier, 145
 - drawItem, 145
 - drawText, 147
 - IdentifierMode, 141
 - identifierMode, 143
 - identifierWidth, 143
 - isChecked, 145
 - isDown, 146
 - itemMode, 142
 - keyPressEvent, 147
 - keyReleaseEvent, 147
 - mousePressEvent, 147
 - mouseReleaseEvent, 147
 - paintEvent, 146
 - pressed, 146
 - QwtLegendItem, 141
 - released, 146
 - setChecked, 145
 - setCurvePen, 144
 - setDown, 146
 - setIdentifierWidth, 143
 - setIdentifierMode, 142
 - setItemMode, 142
 - setSpacing, 143
 - setSymbol, 144
 - setText, 142
 - sizeHint, 145
 - spacing, 143
 - symbol, 144
- QwtLegendItemManager, 147
- QwtLinearColorMap, 148
 - QwtLinearColorMap, 149, 150
- QwtLinearColorMap
 - ~QwtLinearColorMap, 150
 - addColorStop, 151
 - color1, 151
 - color2, 152
 - colorIndex, 152
 - colorStops, 151
 - copy, 150
 - Mode, 149
 - mode, 150
 - operator=, 150
 - QwtLinearColorMap, 149, 150
 - rgb, 152
 - setColorInterval, 151
 - setMode, 150
- QwtLinearScaleEngine, 152

- QwtLinearScaleEngine
 - align, 154
 - autoScale, 153
 - divideScale, 154
 - transformation, 154
- QwtLog10ScaleEngine, 155
- QwtLog10ScaleEngine
 - autoScale, 156
 - divideScale, 156
 - log10, 157
 - pow10, 157
 - transformation, 156
- QwtMagnifier, 157
 - QwtMagnifier, 158
- QwtMagnifier
 - ~QwtMagnifier, 158
 - eventFilter, 162
 - getMouseButton, 160
 - getZoomInKey, 162
 - getZoomOutKey, 162
 - isEnabled, 159
 - keyFactor, 161
 - mouseFactor, 159
 - QwtMagnifier, 158
 - setEnabled, 158
 - setKeyFactor, 161
 - setMouseButton, 159
 - setMouseFactor, 159
 - setWheelButtonState, 160
 - setWheelFactor, 160
 - setZoomInKey, 161
 - setZoomOutKey, 162
 - wheelButtonState, 161
 - wheelFactor, 160
 - widgetKeyPressEvent, 164
 - widgetKeyReleaseEvent, 164
 - widgetMouseMoveEvent, 163
 - widgetMousePressEvent, 163
 - widgetMouseReleaseEvent, 163
 - widgetWheelEvent, 163
- QwtMathMLTextEngine, 164
 - QwtMathMLTextEngine, 165
- QwtMathMLTextEngine
 - ~QwtMathMLTextEngine, 165
 - draw, 166
 - heightForWidth, 166
 - mightRender, 167
 - QwtMathMLTextEngine, 165
 - textMargins, 167
 - textSize, 166
- QwtMetricsMap, 167
- QwtMetricsMap
 - translate, 168
- QwtPainter
 - clip, 174
 - deviceClipping, 171
 - drawEllipse, 173
 - drawLine, 173
 - drawPie, 173
 - drawPoint, 174
 - drawPolygon, 173
 - drawPolyline, 173
 - drawRect, 172
 - drawRoundFrame, 174
 - drawSimpleRichText, 172
 - drawText, 171, 172
 - fillRect, 172
 - metricsMap, 171
 - resetMetricsMap, 170
 - setClipRect, 171
 - setDeviceClipping, 171
 - setMetricsMap, 170
- QwtPanner, 174
 - QwtPanner, 175
- QwtPanner
 - ~QwtPanner, 175
 - cursor, 177
 - eventFilter, 177
 - getAbortKey, 176
 - getMouseButton, 176
 - isEnabled, 176
 - isOrientationEnabled, 177
 - moved, 178
 - orientations, 177
 - paintEvent, 180
 - panned, 178
 - QwtPanner, 175
 - setAbortKey, 176
 - setCursor, 176
 - setEnabled, 176
 - setMouseButton, 176
 - setOrientations, 177
 - widgetKeyPressEvent, 179
 - widgetKeyReleaseEvent, 179
 - widgetMouseMoveEvent, 179
 - widgetMousePressEvent, 178
 - widgetMouseReleaseEvent, 178
- QwtPicker, 180
 - QwtPicker, 186
- QwtPicker
 - ~QwtPicker, 186
 - accept, 194
 - append, 195
 - appended, 193
 - begin, 194
 - changed, 193
 - DisplayMode, 185

- drawRubberBand, 192
- drawTracker, 192
- end, 195
- eventFilter, 191
- isActive, 191
- isEnabled, 190
- move, 195
- moved, 193
- parentWidget, 191
- pickRect, 192
- QwtPicker, 186
- RectSelectionType, 184
- reset, 196
- SizeMode, 185
- resizeMode, 189
- RubberBand, 184
- rubberBand, 187
- rubberBandPen, 189
- selected, 193
- selection, 192
- selectionFlags, 187
- SelectionMode, 184
- SelectionType, 183
- setEnabled, 190
- setSizeMode, 188
- setRubberBand, 187
- setRubberBandPen, 189
- setSelectionFlags, 187
- setTrackerFont, 190
- setTrackerMode, 188
- setTrackerPen, 189
- stateMachine, 199
- stretchSelection, 198
- trackerFont, 190
- trackerMode, 188
- trackerPen, 190
- trackerText, 192
- transition, 194
- widgetKeyPressEvent, 197
- widgetKeyReleaseEvent, 198
- widgetLeaveEvent, 198
- widgetMouseDoubleClickEvent, 196
- widgetMouseMoveEvent, 197
- widgetMousePressEvent, 196
- widgetMouseReleaseEvent, 196
- widgetWheelEvent, 197
- QwtPickerClickPointMachine, 199
- QwtPickerClickPointMachine
 - transition, 200
- QwtPickerClickRectMachine, 200
- QwtPickerClickRectMachine
 - transition, 201
- QwtPickerDragPointMachine, 201
- QwtPickerDragPointMachine
 - transition, 202
- QwtPickerDragRectMachine, 202
- QwtPickerDragRectMachine
 - transition, 202
- QwtPickerMachine, 203
 - QwtPickerMachine, 204
- QwtPickerMachine
 - ~QwtPickerMachine, 204
 - Command, 204
 - QwtPickerMachine, 204
 - reset, 204
 - setState, 204
 - state, 204
 - transition, 204
- QwtPickerPolygonMachine, 205
- QwtPickerPolygonMachine
 - transition, 205
- QwtPlainTextEngine, 206
 - QwtPlainTextEngine, 206
- QwtPlainTextEngine
 - ~QwtPlainTextEngine, 206
 - draw, 207
 - heightForWidth, 207
 - mightRender, 207
 - QwtPlainTextEngine, 206
 - textMargins, 208
 - textSize, 207
- QwtPlot, 208
 - QwtPlot, 212
- QwtPlot
 - ~QwtPlot, 212
 - autoRefresh, 229
 - autoReplot, 213
 - Axis, 212
 - axisAutoScale, 219
 - axisEnabled, 219
 - axisFont, 220
 - axisMaxMajor, 225
 - axisMaxMinor, 226
 - axisScaleDiv, 222
 - axisScaleDraw, 223
 - axisScaleEngine, 218
 - axisStepSize, 221
 - axisTitle, 225
 - axisValid, 230
 - axisWidget, 223, 224
 - canvas, 216
 - canvasBackground, 216
 - canvasLineWidth, 217
 - canvasMap, 217
 - clear, 229
 - drawCanvas, 230
 - drawItems, 230
 - enableAxis, 219

- event, 228
- insertLegend, 226
- invTransform, 217
- legend, 227
- legendChecked, 228
- legendClicked, 228
- legendItemChecked, 230
- legendItemClicked, 229
- LegendPosition, 212
- margin, 214
- minimumSizeHint, 228
- plotLayout, 214
- polish, 227
- print, 213
- printCanvas, 232
- printLegend, 233
- printLegendItem, 231
- printScale, 232
- printTitle, 231
- QwtPlot, 212
- replot, 229
- resizeEvent, 231
- setAutoReplot, 212
- setAxisAutoScale, 219
- setAxisFont, 220
- setAxisLabelAlignment, 224
- setAxisLabelRotation, 224
- setAxisMaxMajor, 226
- setAxisMaxMinor, 225
- setAxisScale, 220
- setAxisScaleDiv, 221
- setAxisScaleDraw, 221
- setAxisScaleEngine, 218
- setAxisTitle, 224, 225
- setCanvasBackground, 216
- setCanvasLineWidth, 216
- setMargin, 214
- setTitle, 215
- sizeHint, 227
- title, 215
- titleLabel, 215
- transform, 218
- updateAxes, 231
- updateLayout, 228
- updateTabOrder, 231
- QwtPlotCanvas, 233
 - QwtPlotCanvas, 235
- QwtPlotCanvas
 - ~QwtPlotCanvas, 235
 - drawCanvas, 237
 - drawContents, 237
 - drawFocusIndicator, 237
 - FocusIndicator, 234
 - focusIndicator, 236
 - invalidatePaintCache, 237
 - PaintAttribute, 234
 - paintCache, 236, 237
 - paintEvent, 237
 - plot, 235
 - QwtPlotCanvas, 235
 - setFocusIndicator, 235
 - setPaintAttribute, 236
 - testPaintAttribute, 236
- QwtPlotCurve, 238
 - QwtPlotCurve, 241
- QwtPlotCurve
 - ~QwtPlotCurve, 242
 - baseline, 249
 - boundingRect, 246
 - brush, 248
 - closePolyline, 255
 - CurveAttribute, 241
 - CurveStyle, 241
 - curveType, 242
 - data, 245
 - dataSize, 245
 - draw, 250, 251
 - drawCurve, 251
 - drawDots, 253
 - drawLines, 252
 - drawSteps, 254
 - drawSticks, 253
 - drawSymbols, 252
 - fillCurve, 254
 - init, 251
 - maxXValue, 246
 - maxYValue, 246
 - minXValue, 246
 - minYValue, 246
 - PaintAttribute, 241
 - pen, 247
 - QwtPlotCurve, 241
 - rtti, 242
 - setBaseline, 248
 - setBrush, 248
 - setCurveAttribute, 246
 - setCurveType, 242
 - setData, 243, 244
 - setPaintAttribute, 242
 - setPen, 247
 - setRawData, 243
 - setStyle, 249
 - setSymbol, 249
 - style, 249
 - symbol, 250
 - testCurveAttribute, 247
 - testPaintAttribute, 243
 - updateLegend, 251

- x, [245](#)
 - y, [245](#)
- QwtPlotDict, [255](#)
 - QwtPlotDict, [256](#)
- QwtPlotDict
 - ~QwtPlotDict, [256](#)
 - autoDelete, [256](#)
 - detachItems, [257](#)
 - itemList, [257](#)
 - QwtPlotDict, [256](#)
 - setAutoDelete, [256](#)
- QwtPlotGrid, [257](#)
 - QwtPlotGrid, [259](#)
- QwtPlotGrid
 - ~QwtPlotGrid, [259](#)
 - draw, [263](#)
 - enableX, [259](#)
 - enableXMin, [260](#)
 - enableY, [259](#)
 - enableYMin, [260](#)
 - majPen, [262](#)
 - minPen, [263](#)
 - QwtPlotGrid, [259](#)
 - rtti, [259](#)
 - setMajPen, [262](#)
 - setMinPen, [263](#)
 - setPen, [262](#)
 - setXDiv, [261](#)
 - setYDiv, [261](#)
 - updateScaleDiv, [263](#)
 - xEnabled, [259](#)
 - xMinEnabled, [260](#)
 - xScaleDiv, [261](#)
 - yEnabled, [260](#)
 - yMinEnabled, [261](#)
 - yScaleDiv, [262](#)
- QwtPlotItem, [264](#)
 - QwtPlotItem, [266](#)
- QwtPlotItem
 - ~QwtPlotItem, [266](#)
 - attach, [266](#)
 - boundingRect, [272](#)
 - detach, [266](#)
 - draw, [272](#)
 - hide, [270](#)
 - invTransform, [275](#)
 - isVisible, [270](#)
 - itemChanged, [271](#)
 - legendItem, [273](#)
 - paintRect, [274](#)
 - plot, [266](#)
 - QwtPlotItem, [266](#)
 - rtti, [267](#)
 - scaleRect, [274](#)
 - setAxis, [270](#)
 - setItemAttribute, [268](#)
 - setRenderHint, [268](#)
 - setTitle, [267](#)
 - setVisible, [270](#)
 - setXAxis, [271](#)
 - setYAxis, [271](#)
 - setZ, [269](#)
 - show, [269](#)
 - testItemAttribute, [268](#)
 - testRenderHint, [269](#)
 - title, [267](#)
 - transform, [274](#)
 - updateLegend, [272](#)
 - updateScaleDiv, [273](#)
 - xAxis, [271](#)
 - yAxis, [271](#)
 - z, [269](#)
- QwtPlotItemList
 - qwt_plot_dict.h, [446](#)
- QwtPlotLayout, [275](#)
 - QwtPlotLayout, [276](#)
- QwtPlotLayout
 - ~QwtPlotLayout, [276](#)
 - activate, [280](#)
 - alignCanvasToScales, [278](#)
 - alignLegend, [282](#)
 - alignScales, [283](#)
 - canvasMargin, [277](#)
 - canvasRect, [282](#)
 - expandLineBreaks, [283](#)
 - invalidate, [281](#)
 - layoutLegend, [282](#)
 - legendPosition, [279](#)
 - legendRatio, [280](#)
 - legendRect, [281](#)
 - margin, [277](#)
 - minimumSizeHint, [280](#)
 - QwtPlotLayout, [276](#)
 - scaleRect, [281](#)
 - setAlignCanvasToScales, [278](#)
 - setCanvasMargin, [277](#)
 - setLegendPosition, [279](#)
 - setLegendRatio, [280](#)
 - setMargin, [277](#)
 - setSpacing, [278](#)
 - spacing, [279](#)
 - titleRect, [281](#)
- QwtPlotMagnifier, [283](#)
 - QwtPlotMagnifier, [284](#)
- QwtPlotMagnifier
 - ~QwtPlotMagnifier, [284](#)
 - canvas, [285](#)
 - isAxisEnabled, [285](#)

- plot, 285
- QwtPlotMagnifier, 284
- rescale, 285
- setAxisEnabled, 284
- QwtPlotMarker, 286
 - QwtPlotMarker, 288
- QwtPlotMarker
 - ~QwtPlotMarker, 288
 - boundingRect, 292
 - draw, 291
 - label, 291
 - labelAlignment, 291
 - linePen, 290
 - LineStyle, 287
 - lineStyle, 289
 - QwtPlotMarker, 288
 - rtti, 288
 - setLabel, 290
 - setLabelAlignment, 291
 - setLinePen, 289
 - setLineStyle, 289
 - setSymbol, 290
 - setValue, 289
 - setXValue, 288
 - setYValue, 288
 - symbol, 290
 - value, 288
 - xValue, 288
 - yValue, 288
- QwtPlotPanner, 292
 - QwtPlotPanner, 293
- QwtPlotPanner
 - ~QwtPlotPanner, 293
 - canvas, 294
 - isAxisEnabled, 294
 - moveCanvas, 295
 - plot, 294
 - QwtPlotPanner, 293
 - setAxisEnabled, 294
- QwtPlotPicker, 295
 - QwtPlotPicker, 296, 297
- QwtPlotPicker
 - append, 302
 - appended, 299
 - canvas, 298
 - end, 302
 - invTransform, 300
 - move, 302
 - moved, 299
 - plot, 298
 - QwtPlotPicker, 296, 297
 - scaleRect, 300
 - selected, 299
 - setAxis, 298
 - trackerText, 301
 - transform, 300, 301
 - xAxis, 298
 - yAxis, 298
- QwtPlotPrintFilter, 303
 - QwtPlotPrintFilter, 304
- QwtPlotPrintFilter
 - ~QwtPlotPrintFilter, 304
 - apply, 306
 - color, 305
 - font, 305
 - Item, 304
 - Options, 304
 - options, 305
 - QwtPlotPrintFilter, 304
 - reset, 306
 - setOptions, 305
- QwtPlotRasterItem, 306
 - QwtPlotRasterItem, 308
- QwtPlotRasterItem
 - ~QwtPlotRasterItem, 308
 - alpha, 309
 - CachePolicy, 308
 - cachePolicy, 309
 - draw, 310
 - invalidateCache, 310
 - QwtPlotRasterItem, 308
 - rasterHint, 310
 - renderImage, 310
 - setAlpha, 309
 - setCachePolicy, 309
- QwtPlotScaleItem, 311
 - QwtPlotScaleItem, 312
- QwtPlotScaleItem
 - ~QwtPlotScaleItem, 312
 - borderDistance, 316
 - draw, 317
 - font, 314
 - isScaleDivFromAxis, 313
 - palette, 314
 - position, 315
 - QwtPlotScaleItem, 312
 - rtti, 313
 - scaleDiv, 313
 - scaleDraw, 315
 - setAlignment, 316
 - setBorderDistance, 316
 - setFont, 314
 - setPalette, 314
 - setPosition, 315
 - setScaleDiv, 313
 - setScaleDivFromAxis, 313
 - setScaleDraw, 314
 - updateScaleDiv, 317

- QwtPlotSpectrogram, 317
 - QwtPlotSpectrogram, 320
- QwtPlotSpectrogram
 - ~QwtPlotSpectrogram, 320
 - boundingRect, 322
 - colorMap, 321
 - contourLevels, 324
 - contourPen, 323
 - contourRasterSize, 326
 - data, 321
 - defaultContourPen, 323
 - DisplayMode, 319
 - draw, 325
 - drawContourLines, 326
 - QwtPlotSpectrogram, 320
 - rasterHint, 322
 - renderContourLines, 326
 - renderImage, 325
 - rtti, 324
 - setColorMap, 321
 - setConrecAttribute, 323
 - setContourLevels, 324
 - setData, 321
 - setDefaultContourPen, 322
 - setDisplayMode, 320
 - testConrecAttribute, 324
 - testDisplayMode, 320
- QwtPlotSvgItem, 327
 - QwtPlotSvgItem, 328
- QwtPlotSvgItem
 - ~QwtPlotSvgItem, 328
 - boundingRect, 329
 - draw, 329
 - loadData, 329
 - loadFile, 329
 - QwtPlotSvgItem, 328
 - render, 330
 - rtti, 330
 - viewBox, 330
- QwtPlotZoomer, 331
 - QwtPlotZoomer, 332, 333
- QwtPlotZoomer
 - accept, 339
 - begin, 339
 - end, 339
 - maxStackDepth, 335
 - minZoomSize, 338
 - move, 336
 - moveBy, 336
 - QwtPlotZoomer, 332, 333
 - rescale, 338
 - setAxis, 335
 - setMaxStackDepth, 335
 - setSelectionFlags, 336
 - setZoomBase, 334
 - widgetKeyPressEvent, 338
 - widgetMouseReleaseEvent, 338
 - zoom, 337
 - zoomBase, 334
 - zoomed, 337
 - zoomRect, 334
 - zoomRectIndex, 336
 - zoomStack, 335
- QwtPolygonFData, 340
 - QwtPolygonFData, 340
- QwtPolygonFData
 - copy, 341
 - operator=, 341
 - QwtPolygonFData, 340
 - size, 341
 - x, 341
 - y, 341
- QwtRasterData, 342
- QwtRasterData
 - contourLines, 344
 - copy, 343
 - discardRaster, 343
 - initRaster, 343
 - range, 344
 - rasterHint, 343
 - value, 344
- QwtRect, 344
 - QwtRect, 345
- QwtRect
 - clip, 345
 - QwtRect, 345
- QwtRichTextEngine, 345
 - QwtRichTextEngine, 346
- QwtRichTextEngine
 - draw, 346
 - heightForWidth, 346
 - mightRender, 347
 - QwtRichTextEngine, 346
 - textMargins, 347
 - textSize, 346
- QwtRoundScaleDraw, 347
 - QwtRoundScaleDraw, 349
- QwtRoundScaleDraw
 - ~QwtRoundScaleDraw, 349
 - center, 350
 - drawBackbone, 351
 - drawLabel, 352
 - drawTick, 351
 - extent, 350
 - moveCenter, 350
 - operator=, 349
 - QwtRoundScaleDraw, 349
 - radius, 349

- setAngleRange, 350
- setRadius, 349
- QwtScaleArithmetic, 352
- QwtScaleArithmetic
 - ceil125, 353
 - ceilEps, 353
 - compareEps, 353
 - floor125, 354
 - floorEps, 353
- QwtScaleDiv, 354
 - QwtScaleDiv, 355
- QwtScaleDiv
 - hBound, 357
 - interval, 356
 - invalidate, 357
 - invert, 358
 - isValid, 358
 - lBound, 356
 - operator!=, 356
 - operator==, 356
 - QwtScaleDiv, 355
 - range, 357
 - setInterval, 356
 - setTicks, 357
 - ticks, 357
- QwtScaleDraw, 358
 - QwtScaleDraw, 360
- QwtScaleDraw
 - ~QwtScaleDraw, 360
 - Alignment, 359
 - alignment, 362
 - boundingLabelRect, 366
 - drawBackbone, 367
 - drawLabel, 368
 - drawTick, 367
 - extent, 361
 - getBorderDistHint, 360
 - labelAlignment, 364
 - labelMatrix, 367
 - labelPosition, 365
 - labelRect, 366
 - labelRotation, 365
 - labelSize, 366
 - length, 363
 - maxLabelHeight, 365
 - maxLabelWidth, 365
 - minLabelDist, 360
 - minLength, 361
 - move, 362
 - operator=, 360
 - orientation, 363
 - pos, 363
 - QwtScaleDraw, 360
 - setAlignment, 363
 - setLabelAlignment, 364
 - setLabelRotation, 364
 - setLength, 362
- QwtScaleEngine, 368
 - QwtScaleEngine, 370
- QwtScaleEngine
 - ~QwtScaleEngine, 370
 - Attribute, 369
 - attributes, 371
 - autoScale, 372
 - buildInterval, 374
 - contains, 373
 - divideInterval, 374
 - divideScale, 373
 - hiMargin, 372
 - loMargin, 372
 - QwtScaleEngine, 370
 - reference, 371
 - setAttribute, 370
 - setAttributes, 370
 - setMargins, 371
 - setReference, 371
 - strip, 373
 - testAttribute, 370
 - transformation, 373
- QwtScaleMap, 374
 - QwtScaleMap, 375
- QwtScaleMap
 - ~QwtScaleMap, 375
 - invTransform, 377
 - p1, 377
 - p2, 378
 - QwtScaleMap, 375
 - s1, 378
 - s2, 378
 - setPaintInterval, 376
 - setPaintXInterval, 376
 - setScaleInterval, 376
 - setTransformation, 376
 - transform, 377
 - transformation, 376
 - xTransform, 377
- QwtScaleTransformation, 378
 - QwtScaleTransformation, 379
- QwtScaleTransformation
 - invXForm, 379
 - QwtScaleTransformation, 379
 - xForm, 379
- QwtScaleWidget, 380
 - QwtScaleWidget, 381
- QwtScaleWidget
 - ~QwtScaleWidget, 381
 - alignment, 388
 - dimForLength, 387

- draw, 389
- drawTitle, 388
- endBorderDist, 383
- getBorderDistHint, 383
- getMinBorderDist, 383
- layoutScale, 389
- margin, 384
- minimumSizeHint, 387
- paintEvent, 389
- penWidth, 385
- QwtScaleWidget, 381
- resizeEvent, 389
- scaleChange, 389
- scaleDivChanged, 382
- scaleDraw, 386
- setAlignment, 388
- setBorderDist, 382
- setLabelAlignment, 386
- setLabelRotation, 387
- setMargin, 384
- setMinBorderDist, 384
- setPenWidth, 385
- setScaleDiv, 385
- setScaleDraw, 386
- setSpacing, 384
- setTitle, 382
- sizeHint, 387
- spacing, 385
- startBorderDist, 383
- title, 382
- titleHeightForWidth, 387
- QwtSimpleCompassRose, 389
 - QwtSimpleCompassRose, 390
- QwtSimpleCompassRose
 - draw, 392
 - drawRose, 392
 - numThornLevels, 391
 - numThorns, 391
 - QwtSimpleCompassRose, 390
 - setNumThornLevels, 391
 - setNumThorns, 391
 - setWidth, 391
- QwtSlider, 393
 - QwtSlider, 395
- QwtSlider
 - BGSTYLE, 395
 - bgStyle, 396
 - borderWidth, 397
 - draw, 399
 - drawSlider, 400
 - drawThumb, 400
 - fontChange, 401
 - getScrollMode, 399
 - getValue, 399
 - layoutSlider, 401
 - minimumSizeHint, 398
 - paintEvent, 400
 - QwtSlider, 395
 - rangeChange, 400
 - resizeEvent, 400
 - scaleChange, 401
 - scaleDraw, 399, 401
 - ScalePos, 395
 - scalePosition, 396
 - setBgStyle, 396
 - setBorderWidth, 397
 - setMargins, 398
 - setOrientation, 396
 - setScaleDraw, 398
 - setScalePosition, 396
 - setThumbLength, 397
 - setThumbWidth, 397
 - sizeHint, 398
 - thumbLength, 397
 - thumbWidth, 397
 - valueChange, 400
 - xyPosition, 401
- QwtSpline, 402
 - QwtSpline, 403
- QwtSpline
 - ~QwtSpline, 403
 - buildNaturalSpline, 404
 - buildPeriodicSpline, 404
 - isValid, 404
 - points, 404
 - QwtSpline, 403
 - reset, 404
 - setPoints, 403
 - value, 404
- QwtSplineCurveFitter, 405
- QwtSymbol, 405
 - QwtSymbol, 407
- QwtSymbol
 - ~QwtSymbol, 407
 - brush, 409
 - draw, 410
 - operator!=, 407
 - operator==, 407
 - pen, 409
 - QwtSymbol, 407
 - setBrush, 408
 - setPen, 408
 - setSize, 407, 408
 - setStyle, 408
 - size, 409
 - Style, 407
 - style, 409
- QwtText, 410

- QwtText, 414
- QwtText
 - ~QwtText, 414
 - backgroundBrush, 418
 - backgroundPen, 417
 - color, 417
 - draw, 420
 - font, 415
 - heightForWidth, 419
 - isEmpty, 415
 - isNull, 415
 - LayoutAttribute, 413
 - operator=, 414
 - PaintAttribute, 413
 - QwtText, 414
 - renderFlags, 416
 - setBackgroundBrush, 418
 - setBackgroundPen, 417
 - setColor, 416
 - setFont, 415
 - setLayoutAttribute, 419
 - setPaintAttribute, 418
 - setRenderFlags, 416
 - setText, 414
 - setTextEngine, 421
 - testLayoutAttribute, 419
 - testPaintAttribute, 418
 - text, 414
 - textEngine, 420, 421
 - TextFormat, 412
 - textSize, 420
 - usedColor, 417
 - usedFont, 415
- QwtTextEngine, 422
 - QwtTextEngine, 422
- QwtTextEngine
 - ~QwtTextEngine, 422
 - draw, 424
 - heightForWidth, 423
 - mightRender, 423
 - QwtTextEngine, 422
 - textMargins, 423
 - textSize, 423
- QwtTextLabel, 424
 - QwtTextLabel, 425
- QwtTextLabel
 - ~QwtTextLabel, 425
 - clear, 426
 - drawContents, 428
 - drawText, 428
 - heightForWidth, 427
 - indent, 426
 - margin, 427
 - minimumSizeHint, 427
 - paintEvent, 428
 - QwtTextLabel, 425
 - setIndent, 426
 - setMargin, 427
 - setText, 426
 - sizeHint, 427
 - text, 426
 - textRect, 428
- QwtThermo, 428
 - QwtThermo, 431
- QwtThermo
 - ~QwtThermo, 431
 - alarmBrush, 433
 - alarmColor, 433
 - alarmEnabled, 434
 - alarmLevel, 434
 - borderWidth, 432
 - draw, 437
 - drawThermo, 437
 - fillBrush, 433
 - fillColor, 433
 - fontChange, 437
 - layoutThermo, 437
 - maxValue, 434
 - minimumSizeHint, 436
 - minValue, 435
 - paintEvent, 437
 - pipeWidth, 434
 - QwtThermo, 431
 - resizeEvent, 438
 - scaleChange, 437
 - scaleDraw, 436, 438
 - scalePosition, 432
 - setAlarmBrush, 433
 - setAlarmColor, 433
 - setAlarmEnabled, 434
 - setAlarmLevel, 434
 - setBorderWidth, 432
 - setFillBrush, 432
 - setFillColor, 433
 - setMargin, 435
 - setMaxValue, 434
 - setMinValue, 435
 - setOrientation, 431
 - setPipeWidth, 434
 - setRange, 435
 - setScaleDraw, 436
 - setScalePosition, 432
 - setValue, 436
 - sizeHint, 435
 - value, 435
- QwtWheel, 438
 - QwtWheel, 439
- QwtWheel

- ~QwtWheel, 439
 - draw, 442
 - drawWheel, 442
 - drawWheelBackground, 443
 - getScrollMode, 443
 - getValue, 443
 - layoutWheel, 442
 - mass, 440
 - minimumSizeHint, 442
 - paintEvent, 442
 - paletteChange, 443
 - QwtWheel, 439
 - resizeEvent, 442
 - setColorArray, 443
 - setInternalBorder, 441
 - setMass, 441
 - setOrientation, 440
 - setTickCnt, 440
 - setTotalAngle, 440
 - setViewAngle, 440
 - setWheelWidth, 441
 - sizeHint, 441
 - valueChange, 443
- radius
- QwtRoundScaleDraw, 349
- range
- QwtRasterData, 344
 - QwtScaleDiv, 357
- rangeChange
- QwtCounter, 73
 - QwtDial, 93
 - QwtDoubleRange, 114
 - QwtSlider, 400
- rasterHint
- QwtPlotRasterItem, 310
 - QwtPlotSpectrogram, 322
 - QwtRasterData, 343
- RectSelectionType
- QwtPicker, 184
- reference
- QwtScaleEngine, 371
- released
- QwtLegendItem, 146
- remove
- QwtLegend, 137
- render
- QwtPlotSvgItem, 330
- renderContourLines
- QwtPlotSpectrogram, 326
- renderFlags
- QwtText, 416
- renderImage
- QwtPlotRasterItem, 310
 - QwtPlotSpectrogram, 325
- replot
- QwtPlot, 229
- rescale
- QwtAbstractScale, 18
 - QwtPlotMagnifier, 285
 - QwtPlotZoomer, 338
- reset
- QwtPicker, 196
 - QwtPickerMachine, 204
 - QwtPlotPrintFilter, 306
 - QwtSpline, 404
- resetMetricsMap
- QwtPainter, 170
- resizeEvent
- QwtDial, 89
 - QwtKnob, 133
 - QwtLegend, 139
 - QwtPlot, 231
 - QwtScaleWidget, 389
 - QwtSlider, 400
 - QwtThermo, 438
 - QwtWheel, 442
- ResizeMode
- QwtPicker, 185
- resizeMode
- QwtPicker, 189
- rgb
- QwtAlphaColorMap, 39
 - QwtColorMap, 52
 - QwtLinearColorMap, 152
- rose
- QwtCompass, 55, 56
- rtti
- QwtPlotCurve, 242
 - QwtPlotGrid, 259
 - QwtPlotItem, 267
 - QwtPlotMarker, 288
 - QwtPlotScaleItem, 313
 - QwtPlotSpectrogram, 324
 - QwtPlotSvgItem, 330
- RubberBand
- QwtPicker, 184
- rubberBand
- QwtPicker, 187
- rubberBandPen
- QwtPicker, 189
- s1
- QwtScaleMap, 378
- s2
- QwtScaleMap, 378
- scaleChange
- QwtAbstractScale, 19

- QwtScaleWidget, 389
- QwtSlider, 401
- QwtThermo, 437
- ScaleComponent
 - QwtAbstractScaleDraw, 21
- scaleContentsRect
 - QwtDial, 88
- scaleDiv
 - QwtAbstractScaleDraw, 22
 - QwtPlotScaleItem, 313
- scaleDivChanged
 - QwtScaleWidget, 382
- scaleDraw
 - QwtDial, 89
 - QwtKnob, 132, 133
 - QwtPlotScaleItem, 315
 - QwtScaleWidget, 386
 - QwtSlider, 399, 401
 - QwtThermo, 436, 438
- scaleEngine
 - QwtAbstractScale, 18
- scaleLabel
 - QwtAnalogClock, 43
 - QwtCompass, 57
 - QwtDial, 93
- scaleMap
 - QwtAbstractScale, 18
 - QwtAbstractScaleDraw, 26
- scaleMaxMajor
 - QwtAbstractScale, 17
- scaleMaxMinor
 - QwtAbstractScale, 17
- ScaleOptions
 - QwtDial, 81
- ScalePos
 - QwtSlider, 395
- scalePosition
 - QwtSlider, 396
 - QwtThermo, 432
- scaleRect
 - QwtPlotItem, 274
 - QwtPlotLayout, 281
 - QwtPlotPicker, 300
- ScrollMode
 - QwtAbstractSlider, 29
- selected
 - QwtPicker, 193
 - QwtPlotPicker, 299
- selection
 - QwtPicker, 192
- selectionFlags
 - QwtPicker, 187
- SelectionMode
 - QwtPicker, 184
- SelectionType
 - QwtPicker, 183
- setAbortKey
 - QwtPanner, 176
- setAbstractScaleDraw
 - QwtAbstractScale, 19
- setAlarmBrush
 - QwtThermo, 433
- setAlarmColor
 - QwtThermo, 433
- setAlarmEnabled
 - QwtThermo, 434
- setAlarmLevel
 - QwtThermo, 434
- setAlignCanvasToScales
 - QwtPlotLayout, 278
- setAlignment
 - QwtPlotScaleItem, 316
 - QwtScaleDraw, 363
 - QwtScaleWidget, 388
- setAlpha
 - QwtPlotRasterItem, 309
- setAngleRange
 - QwtRoundScaleDraw, 350
- setAttribute
 - QwtScaleEngine, 370
- setAttributes
 - QwtScaleEngine, 370
- setAutoDelete
 - QwtPlotDict, 256
- setAutoReplot
 - QwtPlot, 212
- setAutoScale
 - QwtAbstractScale, 16
- setAxis
 - QwtPlotItem, 270
 - QwtPlotPicker, 298
 - QwtPlotZoomer, 335
- setAxisAutoScale
 - QwtPlot, 219
- setAxisEnabled
 - QwtPlotMagnifier, 284
 - QwtPlotPanner, 294
- setAxisFont
 - QwtPlot, 220
- setAxisLabelAlignment
 - QwtPlot, 224
- setAxisLabelRotation
 - QwtPlot, 224
- setAxisMaxMajor
 - QwtPlot, 226
- setAxisMaxMinor
 - QwtPlot, 225
- setAxisScale

- QwtPlot, 220
- setAxisScaleDiv
 - QwtPlot, 221
- setAxisScaleDraw
 - QwtPlot, 221
- setAxisScaleEngine
 - QwtPlot, 218
- setAxisTitle
 - QwtPlot, 224, 225
- setBackgroundBrush
 - QwtText, 418
- setBackgroundPen
 - QwtText, 417
- setBaseline
 - QwtPlotCurve, 248
- setBgStyle
 - QwtSlider, 396
- setBorderDist
 - QwtScaleWidget, 382
- setBorderDistance
 - QwtPlotScaleItem, 316
- setBorderWidth
 - QwtKnob, 131
 - QwtSlider, 397
 - QwtThermo, 432
- setBrush
 - QwtPlotCurve, 248
 - QwtSymbol, 408
- setCachePolicy
 - QwtPlotRasterItem, 309
- setCanvasBackground
 - QwtPlot, 216
- setCanvasLineWidth
 - QwtPlot, 216
- setCanvasMargin
 - QwtPlotLayout, 277
- setChecked
 - QwtLegendItem, 145
- setClipRect
 - QwtPainter, 171
- setColor
 - QwtAlphaColorMap, 39
 - QwtText, 416
- setColorArray
 - QwtWheel, 443
- setColorInterval
 - QwtLinearColorMap, 151
- setColorMap
 - QwtPlotSpectrogram, 321
- setConrecAttribute
 - QwtPlotSpectrogram, 323
- setContourLevels
 - QwtPlotSpectrogram, 324
- setCurrentTime
 - QwtAnalogClock, 43
- setCursor
 - QwtPanner, 176
- setCurveAttribute
 - QwtPlotCurve, 246
- setCurvePen
 - QwtLegendItem, 144
- setCurveType
 - QwtPlotCurve, 242
- setData
 - QwtPlotCurve, 243, 244
 - QwtPlotSpectrogram, 321
- setDefaultContourPen
 - QwtPlotSpectrogram, 322
- setDeviceClipping
 - QwtPainter, 171
- setDisplayMode
 - QwtPlotSpectrogram, 320
- setDisplayPolicy
 - QwtLegend, 136
- setDown
 - QwtLegendItem, 146
- setEditable
 - QwtCounter, 68
- setEnabled
 - QwtMagnifier, 158
 - QwtPanner, 176
 - QwtPicker, 190
- setFillBrush
 - QwtThermo, 432
- setFillColor
 - QwtThermo, 433
- setFocusIndicator
 - QwtPlotCanvas, 235
- setFont
 - QwtPlotScaleItem, 314
 - QwtText, 415
- setFrameShadow
 - QwtDial, 82
- setGeometry
 - QwtDynGridLayout, 118
- setHand
 - QwtAnalogClock, 42
- setIdentifierWidth
 - QwtLegendItem, 143
- setIdentifierMode
 - QwtLegendItem, 142
- setIncSteps
 - QwtCounter, 69
- setIndent
 - QwtTextLabel, 426
- setInternalBorder
 - QwtWheel, 441
- setInterval

- QwtDoubleInterval, 103
- QwtScaleDiv, 356
- setItemAttribute
 - QwtPlotItem, 268
- setItemMode
 - QwtLegendItem, 142
- setKeyFactor
 - QwtMagnifier, 161
- setKeyPattern
 - QwtEventPattern, 125
- setKnobWidth
 - QwtKnob, 131
- setLabel
 - QwtPlotMarker, 290
- setLabelAlignment
 - QwtPlotMarker, 291
 - QwtScaleDraw, 364
 - QwtScaleWidget, 386
- setLabelMap
 - QwtCompass, 56
- setLabelRotation
 - QwtScaleDraw, 364
 - QwtScaleWidget, 387
- setLayoutAttribute
 - QwtText, 419
- setLegendPosition
 - QwtPlotLayout, 279
- setLegendRatio
 - QwtPlotLayout, 280
- setLength
 - QwtScaleDraw, 362
- setLinePen
 - QwtPlotMarker, 289
- setLineStyle
 - QwtPlotMarker, 289
- setLineWidth
 - QwtDial, 83
- setMajPen
 - QwtPlotGrid, 262
- setMargin
 - QwtPlot, 214
 - QwtPlotLayout, 277
 - QwtScaleWidget, 384
 - QwtTextLabel, 427
 - QwtThermo, 435
- setMargins
 - QwtScaleEngine, 371
 - QwtSlider, 398
- setMass
 - QwtAbstractSlider, 31
 - QwtWheel, 441
- setMaxCols
 - QwtDynGridLayout, 116
- setMaxStackDepth
 - QwtPlotZoomer, 335
- setMaxValue
 - QwtCounter, 71
 - QwtDoubleInterval, 105
 - QwtThermo, 434
- setMetricsMap
 - QwtPainter, 170
- setMinBorderDist
 - QwtScaleWidget, 384
- setMinimumExtent
 - QwtAbstractScaleDraw, 25
- setMinPen
 - QwtPlotGrid, 263
- setMinValue
 - QwtCounter, 70
 - QwtDoubleInterval, 105
 - QwtThermo, 435
- setMode
 - QwtDial, 83
 - QwtLinearColorMap, 150
- setMouseButton
 - QwtMagnifier, 159
 - QwtPanner, 176
- setMouseFactor
 - QwtMagnifier, 159
- setMousePattern
 - QwtEventPattern, 124, 125
- setNeedle
 - QwtDial, 87
- setNumButtons
 - QwtCounter, 68
- setNumThornLevels
 - QwtSimpleCompassRose, 391
- setNumThorns
 - QwtSimpleCompassRose, 391
- setOptions
 - QwtPlotPrintFilter, 305
- setOrientation
 - QwtAbstractSlider, 31
 - QwtSlider, 396
 - QwtThermo, 431
 - QwtWheel, 440
- setOrientations
 - QwtPanner, 177
- setOrigin
 - QwtDial, 86
- setPaintAttribute
 - QwtPlotCanvas, 236
 - QwtPlotCurve, 242
 - QwtText, 418
- setPaintInterval
 - QwtScaleMap, 376
- setPaintXInterval
 - QwtScaleMap, 376

- setPalette
 - QwtDialNeedle, 95
 - QwtPlotScaleItem, 314
- setPen
 - QwtPlotCurve, 247
 - QwtPlotGrid, 262
 - QwtSymbol, 408
- setPenWidth
 - QwtDialScaleDraw, 98
 - QwtScaleWidget, 385
- setPeriodic
 - QwtDoubleRange, 111
- setPipeWidth
 - QwtThermo, 434
- setPoints
 - QwtSpline, 403
- setPosition
 - QwtAbstractSlider, 34
 - QwtPlotScaleItem, 315
- setRadius
 - QwtRoundScaleDraw, 349
- setRange
 - QwtDoubleRange, 109
 - QwtThermo, 435
- setRawData
 - QwtPlotCurve, 243
- setReadOnly
 - QwtAbstractSlider, 34
- setReference
 - QwtScaleEngine, 371
- setRenderFlags
 - QwtText, 416
- setRenderHint
 - QwtPlotItem, 268
- setResizeMode
 - QwtPicker, 188
- setRose
 - QwtCompass, 55
- setRubberBand
 - QwtPicker, 187
- setRubberBandPen
 - QwtPicker, 189
- setScale
 - QwtAbstractScale, 15, 16
 - QwtDial, 85
- setScaleArc
 - QwtDial, 85
- setScaleDiv
 - QwtAbstractScaleDraw, 22
 - QwtPlotScaleItem, 313
 - QwtScaleWidget, 385
- setScaleDivFromAxis
 - QwtPlotScaleItem, 313
- setScaleDraw
 - QwtDial, 89
 - QwtKnob, 132
 - QwtPlotScaleItem, 314
 - QwtScaleWidget, 386
 - QwtSlider, 398
 - QwtThermo, 436
- setScaleEngine
 - QwtAbstractScale, 18
- setScaleInterval
 - QwtScaleMap, 376
- setScaleMaxMajor
 - QwtAbstractScale, 17
- setScaleMaxMinor
 - QwtAbstractScale, 17
- setScaleOptions
 - QwtDial, 85
- setScalePosition
 - QwtSlider, 396
 - QwtThermo, 432
- setScaleTicks
 - QwtDial, 86
- setScaleSelectionFlags
 - QwtPicker, 187
 - QwtPlotZoomer, 336
- setSize
 - QwtSymbol, 407, 408
- setSpacing
 - QwtAbstractScaleDraw, 24
 - QwtLegendItem, 143
 - QwtPlotLayout, 278
 - QwtScaleWidget, 384
- setState
 - QwtPickerMachine, 204
- setStep
 - QwtCounter, 70
 - QwtDoubleRange, 111
- setStepButton1
 - QwtCounter, 71
- setStepButton2
 - QwtCounter, 71
- setStepButton3
 - QwtCounter, 71
- setStyle
 - QwtPlotCurve, 249
 - QwtSymbol, 408
- setSymbol
 - QwtKnob, 131
 - QwtLegendItem, 144
 - QwtPlotCurve, 249
 - QwtPlotMarker, 290
- setText
 - QwtLegendItem, 142
 - QwtText, 414
 - QwtTextLabel, 426

- setTextEngine
 - QwtText, 421
- setThumbLength
 - QwtSlider, 397
- setThumbWidth
 - QwtSlider, 397
- setTickCnt
 - QwtWheel, 440
- setTickLength
 - QwtAbstractScaleDraw, 23
- setTicks
 - QwtScaleDiv, 357
- setTime
 - QwtAnalogClock, 43
- setTitle
 - QwtPlot, 215
 - QwtPlotItem, 267
 - QwtScaleWidget, 382
- setTotalAngle
 - QwtKnob, 131
 - QwtWheel, 440
- setTrackerFont
 - QwtPicker, 190
- setTrackerMode
 - QwtPicker, 188
- setTrackerPen
 - QwtPicker, 189
- setTracking
 - QwtAbstractSlider, 30
- setTransformation
 - QwtAbstractScaleDraw, 22
 - QwtScaleMap, 376
- setUpdateTime
 - QwtAbstractSlider, 30
- setValid
 - QwtAbstractSlider, 32
 - QwtDoubleRange, 110
- setValue
 - QwtAbstractSlider, 33
 - QwtCounter, 69
 - QwtDoubleRange, 110
 - QwtPlotMarker, 289
 - QwtThermo, 436
- setViewAngle
 - QwtWheel, 440
- setVisible
 - QwtPlotItem, 270
- setWheelButtonState
 - QwtMagnifier, 160
- setWheelFactor
 - QwtMagnifier, 160
- setWheelWidth
 - QwtWheel, 441
- setWidth
 - QwtDialSimpleNeedle, 101
 - QwtSimpleCompassRose, 391
- setWrapping
 - QwtDial, 84
- setXAxis
 - QwtPlotItem, 271
- setXDiv
 - QwtPlotGrid, 261
- setXValue
 - QwtPlotMarker, 288
- setYAxis
 - QwtPlotItem, 271
- setYDiv
 - QwtPlotGrid, 261
- setYValue
 - QwtPlotMarker, 288
- setZ
 - QwtPlotItem, 269
- setZoomBase
 - QwtPlotZoomer, 334
- setZoomInKey
 - QwtMagnifier, 161
- setZoomOutKey
 - QwtMagnifier, 162
- Shadow
 - QwtDial, 81
- show
 - QwtPlotItem, 269
- showBackground
 - QwtDial, 82
- size
 - QwtArrayData, 46
 - QwtCPointerData, 74
 - QwtData, 77
 - QwtPolygonFData, 341
 - QwtSymbol, 409
- sizeHint
 - QwtArrowButton, 48
 - QwtCounter, 70
 - QwtDial, 88
 - QwtDynGridLayout, 118
 - QwtKnob, 132
 - QwtLegend, 138
 - QwtLegendItem, 145
 - QwtPlot, 227
 - QwtScaleWidget, 387
 - QwtSlider, 398
 - QwtTextLabel, 427
 - QwtThermo, 435
 - QwtWheel, 441
- sliderMoved
 - QwtAbstractSlider, 34
- sliderPressed
 - QwtAbstractSlider, 34

- sliderReleased
 - QwtAbstractSlider, 34
- spacing
 - QwtAbstractScaleDraw, 24
 - QwtLegendItem, 143
 - QwtPlotLayout, 279
 - QwtScaleWidget, 385
- startBorderDist
 - QwtScaleWidget, 383
- state
 - QwtPickerMachine, 204
- stateMachine
 - QwtPicker, 199
- step
 - QwtCounter, 70
 - QwtDoubleRange, 111
- stepButton1
 - QwtCounter, 71
- stepButton2
 - QwtCounter, 71
- stepButton3
 - QwtCounter, 71
- stepChange
 - QwtDoubleRange, 114
- stopMoving
 - QwtAbstractSlider, 30
- stretchGrid
 - QwtDynGridLayout, 119
- stretchSelection
 - QwtPicker, 198
- strip
 - QwtScaleEngine, 373
- Style
 - QwtCompassMagnetNeedle, 60
 - QwtCompassWindArrow, 64
 - QwtDialSimpleNeedle, 100
 - QwtSymbol, 407
- style
 - QwtPlotCurve, 249
 - QwtSymbol, 409
- Symbol
 - QwtKnob, 130
- symbol
 - QwtKnob, 132
 - QwtLegendItem, 144
 - QwtPlotCurve, 250
 - QwtPlotMarker, 290
- symmetrize
 - QwtDoubleInterval, 107
- testAttribute
 - QwtScaleEngine, 370
- testConrecAttribute
 - QwtPlotSpectrogram, 324
- testCurveAttribute
 - QwtPlotCurve, 247
- testDisplayMode
 - QwtPlotSpectrogram, 320
- testItemAttribute
 - QwtPlotItem, 268
- testLayoutAttribute
 - QwtText, 419
- testPaintAttribute
 - QwtPlotCanvas, 236
 - QwtPlotCurve, 243
 - QwtText, 418
- testRenderHint
 - QwtPlotItem, 269
- text
 - QwtText, 414
 - QwtTextLabel, 426
- textEngine
 - QwtText, 420, 421
- TextFormat
 - QwtText, 412
- textMargins
 - QwtMathMLTextEngine, 167
 - QwtPlainTextEngine, 208
 - QwtRichTextEngine, 347
 - QwtTextEngine, 423
- textRect
 - QwtTextLabel, 428
- textSize
 - QwtMathMLTextEngine, 166
 - QwtPlainTextEngine, 207
 - QwtRichTextEngine, 346
 - QwtText, 420
 - QwtTextEngine, 423
- thumbLength
 - QwtSlider, 397
- thumbWidth
 - QwtSlider, 397
- tickLabel
 - QwtAbstractScaleDraw, 27
- tickLength
 - QwtAbstractScaleDraw, 24
- ticks
 - QwtScaleDiv, 357
- timerEvent
 - QwtAbstractSlider, 35
- title
 - QwtPlot, 215
 - QwtPlotItem, 267
 - QwtScaleWidget, 382
- titleHeightForWidth
 - QwtScaleWidget, 387
- titleLabel
 - QwtPlot, 215

- titleRect
 - QwtPlotLayout, 281
- totalAngle
 - QwtKnob, 131
- trackerFont
 - QwtPicker, 190
- trackerMode
 - QwtPicker, 188
- trackerPen
 - QwtPicker, 190
- trackerText
 - QwtPicker, 192
 - QwtPlotPicker, 301
- transform
 - QwtPlot, 218
 - QwtPlotItem, 274
 - QwtPlotPicker, 300, 301
 - QwtScaleMap, 377
- transformation
 - QwtLinearScaleEngine, 154
 - QwtLog10ScaleEngine, 156
 - QwtScaleEngine, 373
 - QwtScaleMap, 376
- transition
 - QwtPicker, 194
 - QwtPickerClickPointMachine, 200
 - QwtPickerClickRectMachine, 201
 - QwtPickerDragPointMachine, 202
 - QwtPickerDragRectMachine, 202
 - QwtPickerMachine, 204
 - QwtPickerPolygonMachine, 205
- translate
 - QwtMetricsMap, 168
- unite
 - QwtDoubleInterval, 106
- updateAxes
 - QwtPlot, 231
- updateLayout
 - QwtPlot, 228
- updateLegend
 - QwtPlotCurve, 251
 - QwtPlotItem, 272
- updateMask
 - QwtDial, 90
- updateScale
 - QwtDial, 93
- updateScaleDiv
 - QwtPlotGrid, 263
 - QwtPlotItem, 273
 - QwtPlotScaleItem, 317
- updateTabOrder
 - QwtPlot, 231
- usedColor
 - QwtText, 417
- usedFont
 - QwtText, 415
- value
 - QwtCounter, 72
 - QwtDoubleRange, 110
 - QwtPlotMarker, 288
 - QwtRasterData, 344
 - QwtSpline, 404
 - QwtThermo, 435
- valueChange
 - QwtAbstractSlider, 35
 - QwtDial, 93
 - QwtDoubleRange, 114
 - QwtSlider, 400
 - QwtWheel, 443
- valueChanged
 - QwtAbstractSlider, 34
 - QwtCounter, 72
- viewBox
 - QwtPlotSvgItem, 330
- wheelButtonState
 - QwtMagnifier, 161
- wheelEvent
 - QwtAbstractSlider, 36
- wheelFactor
 - QwtMagnifier, 160
- widgetKeyPressEvent
 - QwtMagnifier, 164
 - QwtPanner, 179
 - QwtPicker, 197
 - QwtPlotZoomer, 338
- widgetKeyReleaseEvent
 - QwtMagnifier, 164
 - QwtPanner, 179
 - QwtPicker, 198
- widgetLeaveEvent
 - QwtPicker, 198
- widgetMouseDoubleClickEvent
 - QwtPicker, 196
- widgetMouseMoveEvent
 - QwtMagnifier, 163
 - QwtPanner, 179
 - QwtPicker, 197
- widgetMousePressEvent
 - QwtMagnifier, 163
 - QwtPanner, 178
 - QwtPicker, 196
- widgetMouseReleaseEvent
 - QwtMagnifier, 163
 - QwtPanner, 178
 - QwtPicker, 196

- QwtPlotZoomer, 338
- widgetWheelEvent
 - QwtMagnifier, 163
 - QwtPicker, 197
- width
 - QwtDialSimpleNeedle, 101
 - QwtDoubleInterval, 104
- wrapping
 - QwtDial, 84
- x
 - QwtArrayData, 46
 - QwtCPointerData, 74
 - QwtData, 78
 - QwtPlotCurve, 245
 - QwtPolygonFData, 341
- xAxis
 - QwtPlotItem, 271
 - QwtPlotPicker, 298
- xData
 - QwtArrayData, 47
 - QwtCPointerData, 75
- xEnabled
 - QwtPlotGrid, 259
- xForm
 - QwtScaleTransformation, 379
- xMinEnabled
 - QwtPlotGrid, 260
- xScaleDiv
 - QwtPlotGrid, 261
- xTransform
 - QwtScaleMap, 377
- xValue
 - QwtPlotMarker, 288
- xyPosition
 - QwtSlider, 401
- y
 - QwtArrayData, 46
 - QwtCPointerData, 75
 - QwtData, 78
 - QwtPlotCurve, 245
 - QwtPolygonFData, 341
- yAxis
 - QwtPlotItem, 271
 - QwtPlotPicker, 298
- yData
 - QwtArrayData, 47
 - QwtCPointerData, 75
- yEnabled
 - QwtPlotGrid, 260
- yMinEnabled
 - QwtPlotGrid, 261
- yScaleDiv
 - QwtPlotGrid, 262
- yValue
 - QwtPlotMarker, 288
- z
 - QwtPlotItem, 269
- zoom
 - QwtPlotZoomer, 337
- zoomBase
 - QwtPlotZoomer, 334
- zoomed
 - QwtPlotZoomer, 337
- zoomRect
 - QwtPlotZoomer, 334
- zoomRectIndex
 - QwtPlotZoomer, 336
- zoomStack
 - QwtPlotZoomer, 335