EPICS QT Framework

2.6.0

Generated by Doxygen 1.7.4

Mon Jul 1 2013 10:02:07

Contents

Chapter 1

QE framework - EPICS aware Qt Widgets and data access classes

- QE is a layered software framework for accessing EPICS data using Channel Access on a range of platforms.

- The QE framework provides object oriented C++ access to control systems using EPICS (Experimental Physics and Industrial Control System). It is based on Qt, a widely used cross-platform application development framework.

- GUI or console based applications can be written that use QE at several levels. QE includes Qt plugin libraries, EPICS aware widgets, data formatting classes, and classes for accessing raw EPICS data in a Qt friendly way.

- QE also includes an application - QEgui - for displaying forms produced by the Qt development tool 'Designer'. Using this application a complete EPICS GUI system can be generated without writing any code. A GUI system produced in this way can interact with existing EPICS display tools such as EDM.

- QE handles much of the complexities of Channel Access including initiating and managing a channel. Applications using QE can interact with Channel Access using Qt based classes and data types. Channel Access updates are delivered using Qt's signals and slots mechanism.

1.1 Documentation

Support documents can be found in the documentation section of the epicsqt sourceforge project. The framework download (available on the epicsqt sourceforge homepage) also includes this documentation as well as full Doxygen generated documentation of all the epicsqt classes and widgets.

1.2 License

epicsqt is distributed under the terms of the GNU General Public License.

1.3 Platforms

epicsqt might be usable in all environments where you find Qt. It is compatible with Qt >= 4.4.

1.4 Screenshots

- ASgui screen shots
- other applications using epicsqt widgets
- Qt Designer
- Qt Creator

Screenshots are only available in the HTML docs.

1.5 Downloads

Stable releases and development snapshots are available at the epicsqt project page.

For getting a development snapshot from the SVN repository:

```
svn svn co https://epicsqt.svn.sourceforge.net/svnroot/epicsqt epicsqt
```

Alternativly, get a packaged file (epicsqt.tar.gz) from the epicsqt repository site.

1.6 Installation

Read QE_GettingStarted.pdf in the documentation for setting up an enviroment for building or using the epicsqt framework.

To build the framework, open epicsqt.pro in QtCreator, ensure shaddow build is turned off, and hit build.

The resultant library libQEPlugin.so will need to be installed or referenced up according to how it is to be used - see QE_GettingStarted.pdf for details.

Any Qt specific queries? start at the Qt Project

1.7 Support

Visit the sourceforge epicsqt support page for assistance.

1.8 Related Projects

Qwt, The core of a Channel Access aware plotting widget.

1.9 Credits:

Authors:

Andrew Rhyder, Anthony Owen, Glenn Jackson

Project admin:

Andrew Rhyder <andrew.rhyder@synchrotron.org.au>

Chapter 2

GNU General Public License

The EPICS QT Framework is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The EPICS QT Framework 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 General Public License for more details.

You should have received a copy of the GNU General Public License along with the EPICS QT Framework.

If not, see "http://www.gnu.org/licenses/

Chapter 3

ASgui screen shots

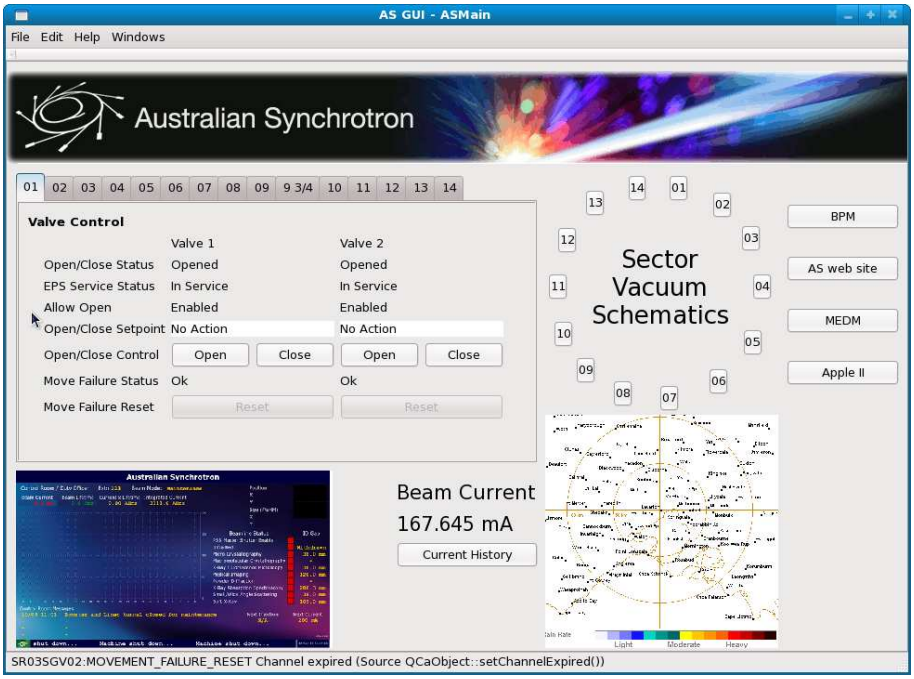


Figure 3.1: Australian Synchrotron mock up

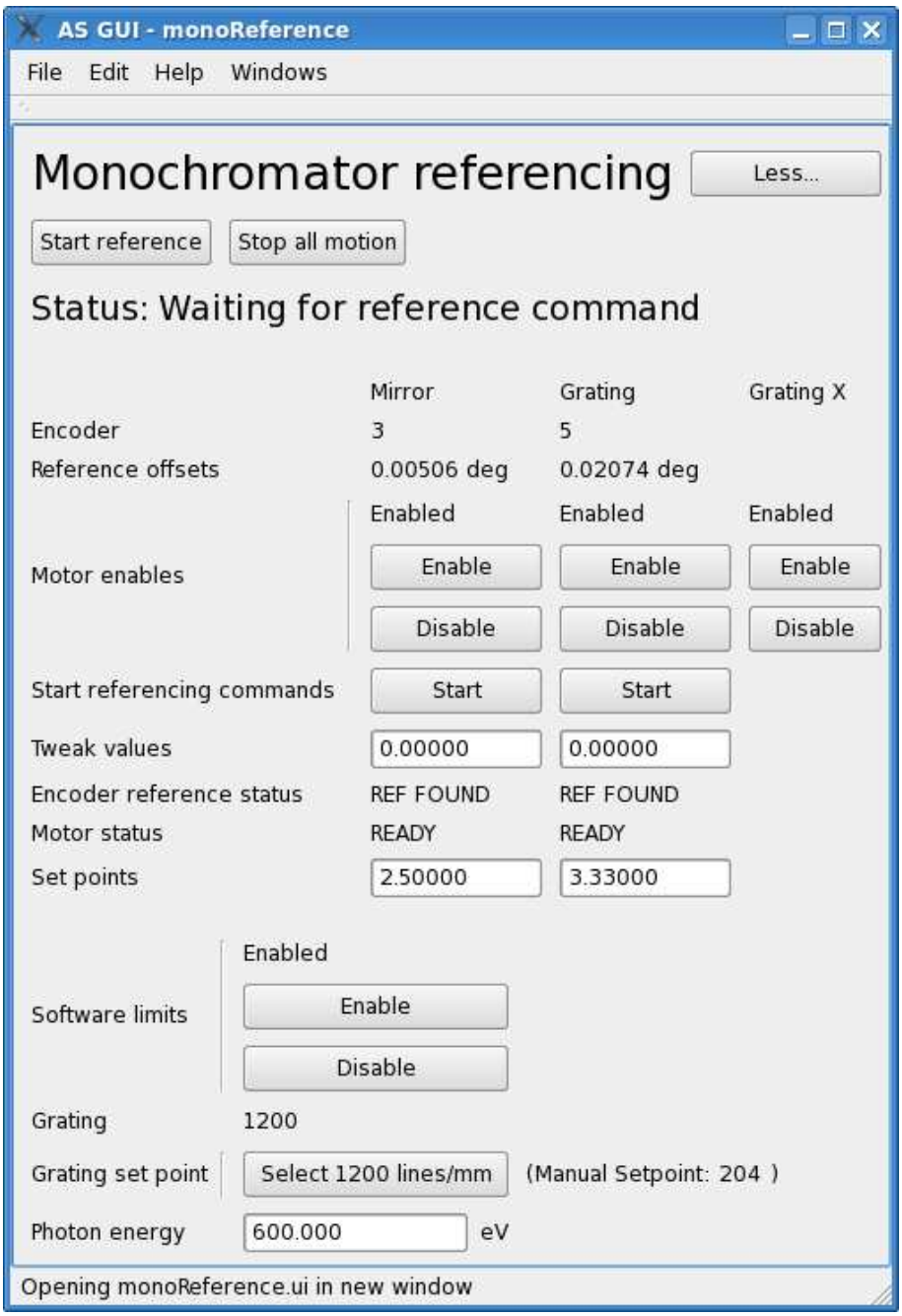


Figure 3.2: Monochromator referencing

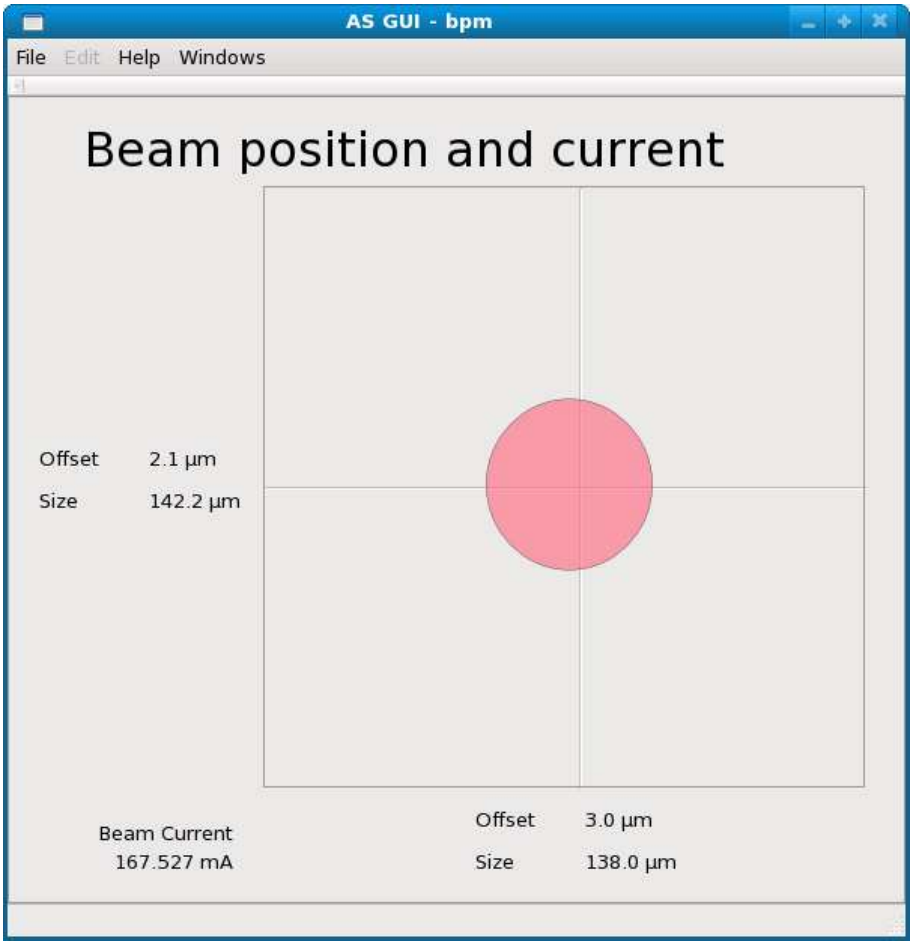


Figure 3.3: Beam position monitor

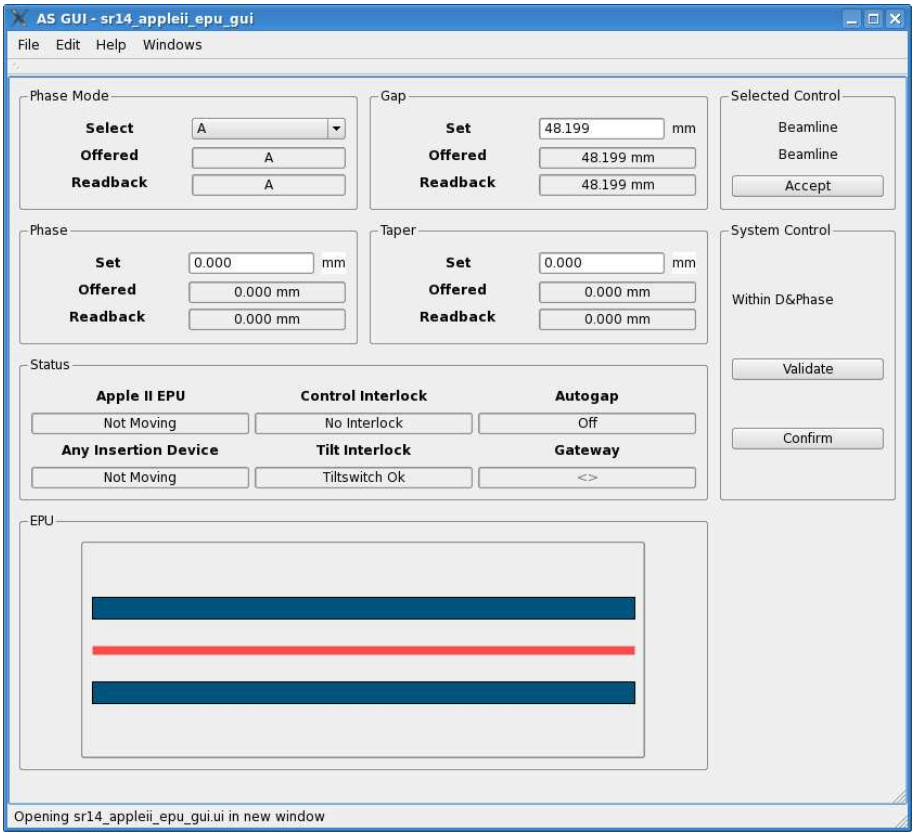


Figure 3.4: Insertion device

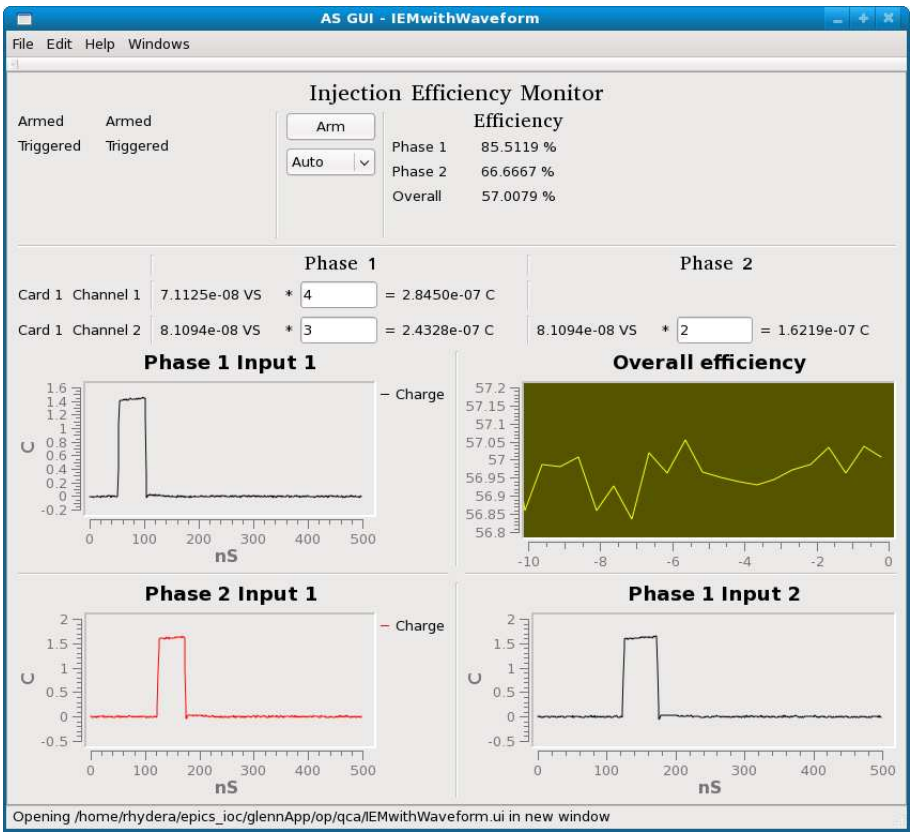


Figure 3.5: Injection efficiency monitor

Chapter 4

other applications using epicsqt widgets

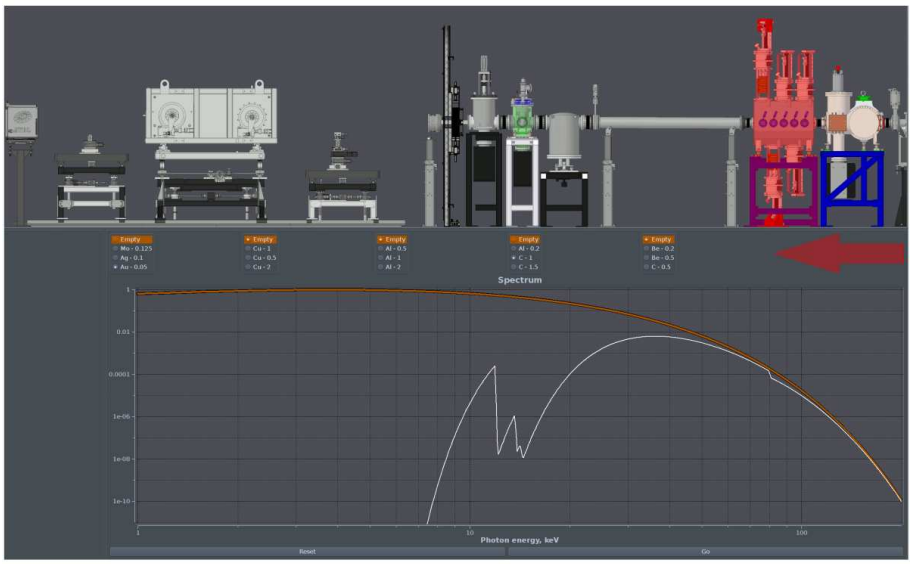


Figure 4.1: Medical Imaging beamline

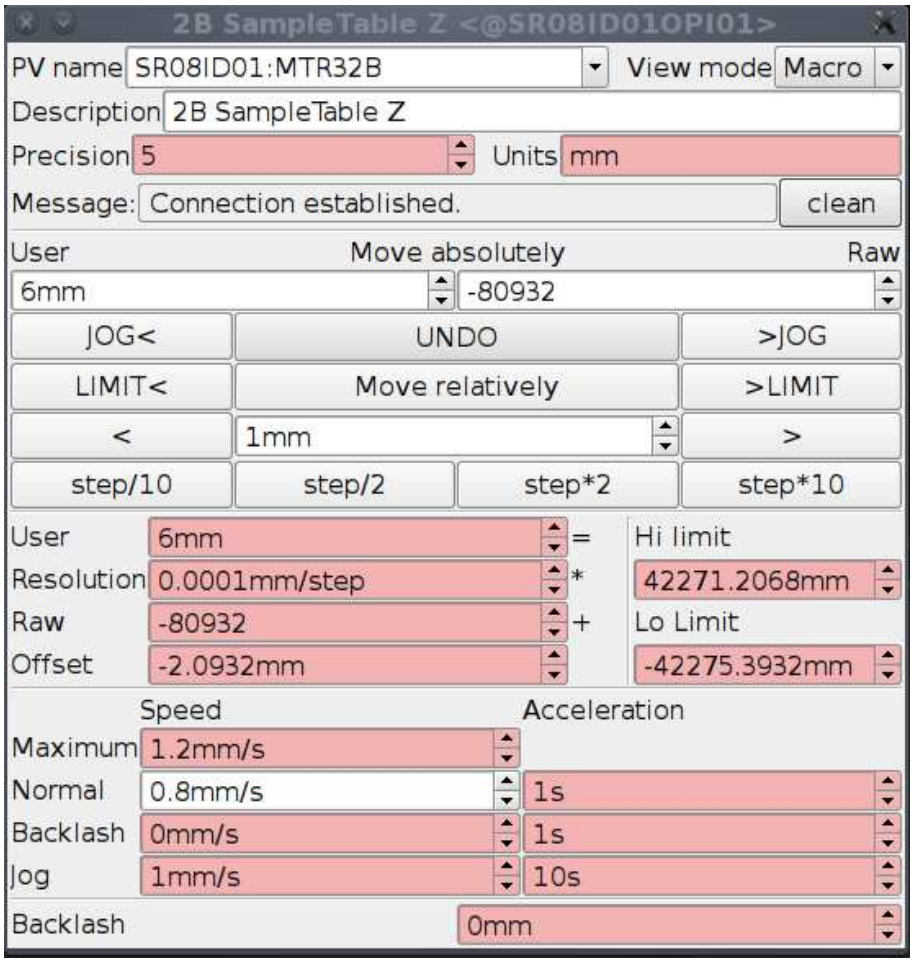


Figure 4.2: Motor controller

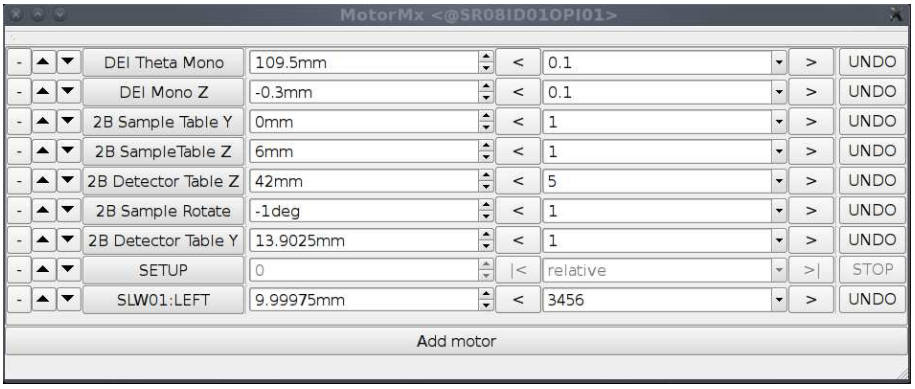


Figure 4.3: Motor controller

Chapter 5

Qt Designer

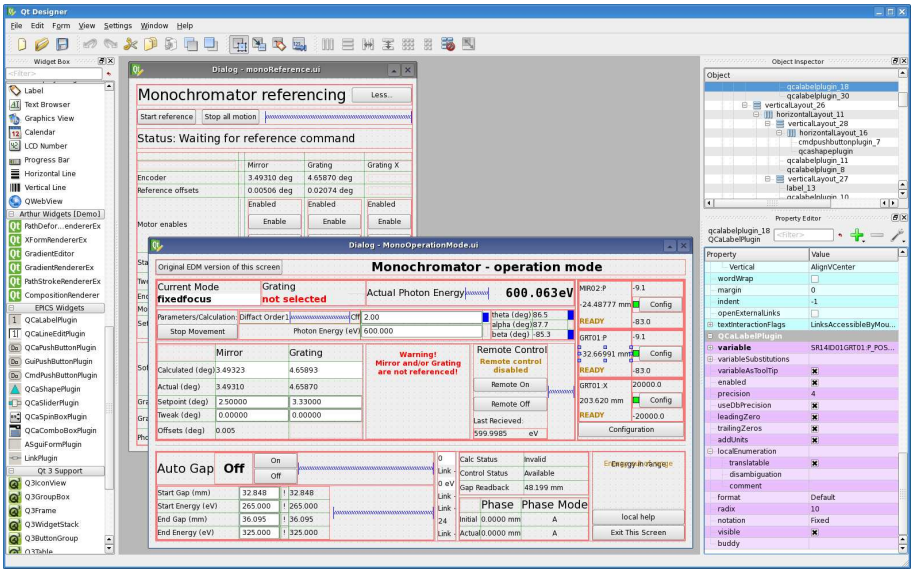


Figure 5.1: Editing multiple GUIs

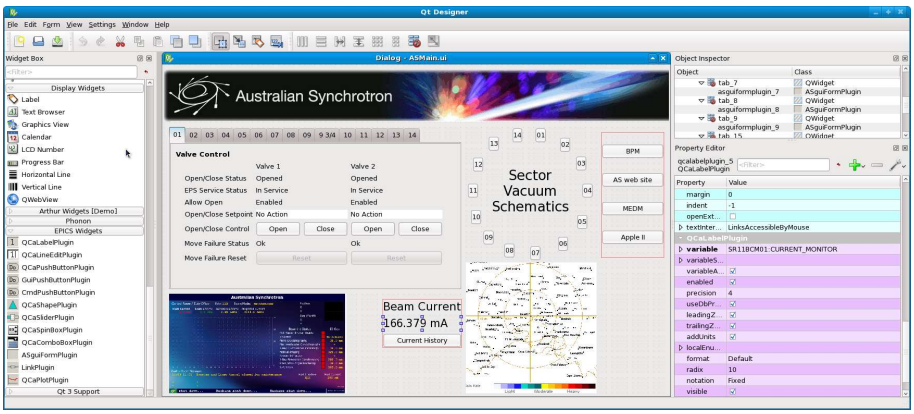


Figure 5.2: Editing a GUI

Chapter 6

Qt Creator

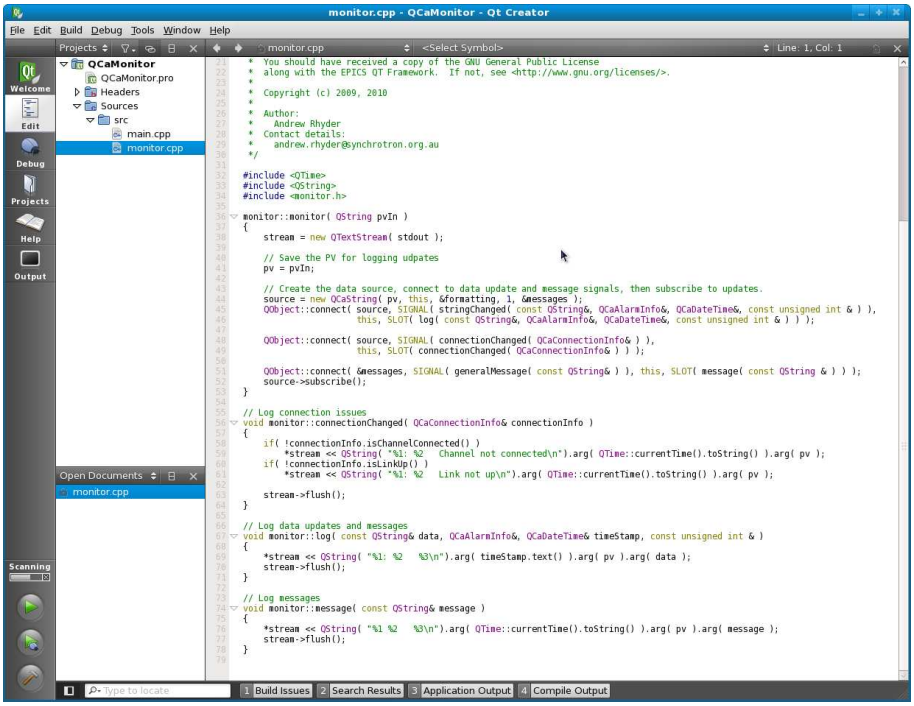


Figure 6.1: Application using epicsqt data source classes

Chapter 7

Class Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

Class Documentation

9.1 _Field Class Reference

Public Member Functions

- QEWidget * **getWidget** ()
- void **setWidget** (QString *pValue)
- QString **getName** ()
- void **setName** (QString pValue)
- QString **getProcessVariable** ()
- void **setProcessVariable** (QString pValue)
- void **setJoin** (bool pValue)
- bool **getJoin** ()
- int **getType** ()
- void **setType** (int pValue)
- QString **getGroup** ()
- void **setGroup** (QString pValue)
- QString **getVisible** ()
- void **setVisible** (QString pValue)
- QString **getEditable** ()
- void **setEditable** (QString pValue)
- bool **getVisibility** ()
- void **setVisibility** (bool pValue)

Public Attributes

- QEWidget * **qCaWidget**

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.2 _Item Class Reference

Public Member Functions

- void **setName** (QString pValue)
- QString **getName** ()
- void **setSubstitution** (QString pValue)
- QString **getSubstitution** ()
- void **setVisible** (QString pValue)
- QString **getVisible** ()

Public Attributes

- QList< _Field * > **fieldList**

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.3 _QDialogItem Class Reference

Public Member Functions

- **_QDialogItem** (QWidget *pParent=0, QString pItemName="", QString pGroupName="", QList< _Field * > *pCurrentFieldList=0, Qt::WindowFlags pF=0)

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.4 _QPushButtonGroup Class Reference

Public Slots

- void **buttonGroupClicked** ()

Public Member Functions

- **_QPushButtonGroup** (QWidget *pParent=0, QString pItemName="", QString pGroupName="", QList< _Field * > *pCurrentFieldList=0)
- void **mouseReleaseEvent** (QMouseEvent *qMouseEvent)

- void **keyPressEvent** (QKeyEvent ∗pKeyEvent)
- void **showDialogGroup** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.5 _QTableWidgetFileBrowser Class Reference

Public Member Functions

- **_QTableWidgetFileBrowser** (QWidget ∗pParent=0)
- void **refreshSize** ()
- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

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

- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.6 _QTableWidgetLog Class Reference

Public Member Functions

- **_QTableWidgetLog** (QWidget ∗pParent=0)
- void **refreshSize** ()
- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

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

- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.h
- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.cpp

9.7 _QTableWidgetScript Class Reference

Public Member Functions

- **_QTableWidgetScript** (QWidget ∗pParent=0)
- void **refreshSize** ()

- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

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

- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.h
- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.8 QEAnalogIndicator::Band Struct Reference

Public Attributes

- double **lower**
- double **upper**
- QColor **colour**

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h

9.9 QEAnalogIndicator::BandList Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h

9.10 ChartState Class Reference

Public Member Functions

- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

Public Attributes

- bool **isNormalVideo**
- QEStripChartNames::ChartTimeModes **chartTimeMode**
- QEStripChartNames::YScaleModes **yScaleMode**
- QEStripChartNames::ChartYRanges **chartYScale**
- double **yMinimum**
- double **yMaximum**
- int **duration**

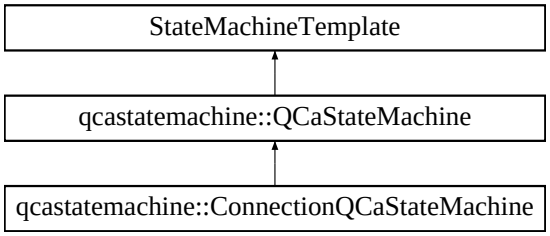
- Qt::TimeSpec **timeZoneSpec**
- QDateTime **endDateTime**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.11 qcastatemachine::ConnectionQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ConnectionQCaStateMachine:



Public Member Functions

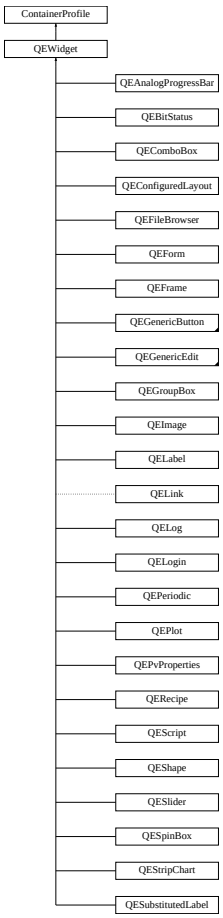
- **ConnectionQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.12 ContainerProfile Class Reference

Inheritance diagram for ContainerProfile:



Public Member Functions

- void **takeLocalCopy** ()
- void **setupProfile** (QObject ∗guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)
- void **setupLocalProfile** (QObject ∗guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)
- void **updateConsumers** (QObject ∗guiLaunchConsumerIn)
- QObject ∗ **replaceGuiLaunchConsumer** (QObject ∗newGuiLaunchConsumerIn)

- void **addMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removeMacroSubstitutions** ()
- void **addPriorityMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removePriorityMacroSubstitutions** ()
- QObject ∗ **getGuiLaunchConsumer** ()
- QString **getPath** ()
- QStringList **getPathList** ()
- QString **getParentPath** ()

- void **setPublishedParentPath** (QString publishedParentPathIn)
- QString **getMacroSubstitutions** ()
- bool **isProfileDefined** ()
- bool **areUserLevelPasswordsSet** ()
- QStringList **getEnvPathList** ()
- QString **getUserLevelPassword** (userLevelTypes::userLevels level)
- void **setUserLevelPassword** (userLevelTypes::userLevels level, QString passwordIn)
- void **addContainedWidget** (QEWidget *containedWidget)
- QEWidget * **getNextContainedWidget** ()
- void **removeContainedWidget** (QEWidget *containedWidget)
- unsigned int **getMessageFormId** ()
- unsigned int **getPublishedMessageFormId** ()
- void **setPublishedMessageFormId** (unsigned int publishedMessageFormIdIn)
- bool **setDontActivateYet** (bool dontActivateIn)
- bool **getDontActivateYet** ()
- void **releaseProfile** ()
- void **publishOwnProfile** ()
- void **setUserLevel** (userLevelTypes::userLevels level)
- userLevelTypes::userLevels **getUserLevel** ()
- virtual void **userLevelChangedGeneral** (userLevelTypes::userLevels)
- PersistanceManager * **getPersistanceManager** ()

Static Public Member Functions

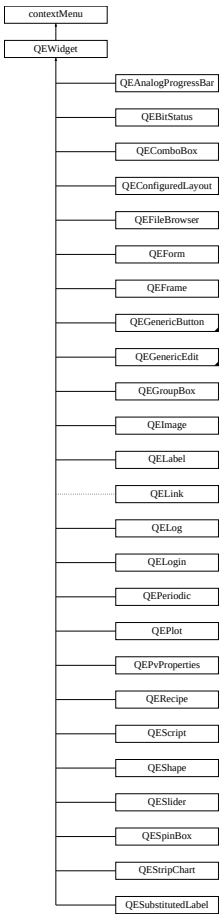
- static QChar **platformSeperator** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h
- /tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp

9.13 contextMenu Class Reference

Inheritance diagram for contextMenu:



Public Types

- enum **contextMenuOptions** {

  CM_NONE, **CM_COPY_VARIABLE**, **CM_COPY_DATA**, **CM_PASTE**,

  CM_DRAG_VARIABLE, **CM_DRAG_DATA**, **CM_SPECIFIC_WIDGETS_START_-HERE** }

Public Member Functions

- void **addContextMenuToWidget** (QWidget ∗w)
- bool **isDraggingVariable** ()
- QMenu ∗ **getContextMenu** ()
- virtual QString **copyVariable** ()
- virtual QVariant **copyData** ()
- virtual void **paste** (QVariant)

Friends

- class contextMenuObject

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

- /tmp/epicsqt/trunk/framework/widgets/include/contextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/src/contextMenu.cpp

9.14 contextMenuObject Class Reference

Public Slots

- void **contextMenuTriggered** (QAction ∗selectedItem)
- void **showContextMenu** (const QPoint &pos)
- void **setChecked** ()

Public Member Functions

- void **addContextMenuToWidget** (QWidget ∗w)
- void **manageChecked** (bool draggingVariable)
- void **setMenu** (contextMenu ∗menuIn)
- bool **isDraggingVariable** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/contextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/src/contextMenu.cpp

9.15 QEPeriodic::elementInfoStruct Struct Reference

Public Attributes

- unsigned int **number**
- double **atomicWeight**
- QString **name**
- QString **symbol**
- double **meltingPoint**
- double **boilingPoint**
- double **density**
- unsigned int **group**
- double **ionizationEnergy**
- unsigned int **tableRow**

- unsigned int **tableCol**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.16 flipRotateMenu Class Reference

Public Member Functions

- **flipRotateMenu** (QWidget *parent=0)
- imageContextMenu::imageContextMenuOptions **getFlipRotate** (const QPoint &pos)

- void **setChecked** (const int rotation, const bool flipH, const bool flipV)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/flipRotateMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/flipRotateMenu.cpp

9.17 imageContextMenu Class Reference

Public Types

- enum **imageContextMenuOptions** {

  ICM_NONE = contextMenu::CM_SPECIFIC_WIDGETS_START_HERE, **ICM_SAVE**, **ICM_PAUSE**, **ICM_ENABLE_TIME**,

  ICM_ENABLE_CURSOR_PIXEL, **ICM_ENABLE_CONTRAST_REVERSAL**, **ICM_ENABLE_VERT**, **ICM_ENABLE_HOZ**,

  ICM_ENABLE_AREA, **ICM_ENABLE_LINE**, **ICM_ENABLE_TARGET**, **ICM_DISPLAY_BUTTON_BAR**,

  ICM_DISPLAY_BRIGHTNESS_CONTRAST, **ICM_ZOOM_SELECTED**, **ICM_ZOOM_FIT**, **ICM_ZOOM_10**,

  ICM_ZOOM_25, **ICM_ZOOM_50**, **ICM_ZOOM_75**, **ICM_ZOOM_100**,

  ICM_ZOOM_150, **ICM_ZOOM_200**, **ICM_ZOOM_300**, **ICM_ZOOM_400**,

  ICM_ROTATE_NONE, **ICM_ROTATE_RIGHT**, **ICM_ROTATE_LEFT**, **ICM_ROTATE_180**,

  ICM_FLIP_HORIZONTAL, **ICM_FLIP_VERTICAL**, **ICM_SELECT_PAN**, **ICM_SELECT_HSLICE**,

  ICM_SELECT_VSLICE, **ICM_SELECT_AREA1**, **ICM_SELECT_AREA2**, **ICM_SELECT_AREA3**,

ICM_SELECT_AREA4, **ICM_SELECT_PROFILE**, **ICM_SELECT_TARGET**, **ICM_-SELECT_BEAM**,

ICM_CLEAR_MARKUP, **ICM_THICKNESS_ONE_MARKUP**, **ICM_THICKNESS_-SELECT_MARKUP**, **ICM_COPY_PLOT_DATA** }

Public Member Functions

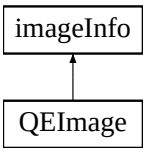
- **imageContextMenu** (QWidget ∗parent=0)
- void **getContextMenuOption** (const QPoint &, imageContextMenuOptions ∗option, bool ∗checked)
- void **addMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)
- void **addOptionMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageContextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageContextMenu.cpp

9.18 imageInfo Class Reference

Inheritance diagram for imageInfo:



Public Member Functions

- void **showInfo** (bool show)
- QLayout ∗ **getInfoWidget** ()
- void **infoShow** (const bool show)
- void **infoUpdateTarget** ()
- void **infoUpdateTarget** (const int x, const int y)
- void **infoUpdateBeam** ()
- void **infoUpdateBeam** (const int x, const int y)
- void **infoUpdateVertProfile** ()
- void **infoUpdateVertProfile** (const int x, const unsigned int thickness)
- void **infoUpdateHozProfile** ()
- void **infoUpdateHozProfile** (const int y, const unsigned int thickness)
- void **infoUpdateProfile** ()

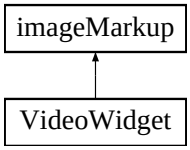
- void **infoUpdateProfile** (const QPoint start, const QPoint end, const unsigned int thickness)
- void **infoUpdateRegion** (const unsigned int region)
- void **infoUpdateRegion** (const unsigned int region, const int x1, const int y1, const int x2, const int y2)
- void **infoUpdatePixel** ()
- void **infoUpdatePixel** (const QPoint pos, int value)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageInfo.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageInfo.cpp

9.19 imageMarkup Class Reference

Inheritance diagram for imageMarkup:



Public Types

- enum **markupIds** {

  MARKUP_ID_REGION1, **MARKUP_ID_REGION2**, **MARKUP_ID_REGION3**, **MARKUP_-ID_REGION4**,

  MARKUP_ID_H_SLICE, **MARKUP_ID_V_SLICE**, **MARKUP_ID_LINE**, **MARKUP_-ID_TARGET**,

  MARKUP_ID_BEAM, **MARKUP_ID_TIMESTAMP**, **MARKUP_ID_COUNT**, **MARKUP_-ID_NONE** }

Public Member Functions

- void **setShowTime** (bool visibleIn)
- bool **getShowTime** ()
- markupIds **getMode** ()
- void **setMode** (markupIds modeIn)
- void **setMarkupColor** (markupIds mode, QColor markupColorIn)
- QColor **getMarkupColor** (markupIds mode)
- bool **showMarkupMenu** (const QPoint &pos, const QPoint &globalPos)
- void **markupRegionValueChange** (int areaIndex, QRect area)

- QCursor **getCircleCursor** ()
- QCursor **getTargetCursor** ()
- QCursor **getVLineCursor** ()
- QCursor **getHLineCursor** ()
- QCursor **getLineCursor** ()
- QCursor **getRegionCursor** ()
- virtual void **markupSetCursor** (QCursor cursor)=0

Public Attributes

- QImage * **markupImage**
- QVector< markupItem * > **items**
- QPoint **grabOffset**
- bool **markupAreasStale**
- QFont **legendFont**
- QFontMetrics * **legendFontMetrics**

Protected Member Functions

- bool **anyVisibleMarkups** ()
- QVector< QRect > & **getMarkupAreas** ()
- QCursor **getDefaultMarkupCursor** ()
- void **setMarkupTime** (QCaDateTime &time)
- bool **markupMousePressEvent** (QMouseEvent *event, bool panning)
- bool **markupMouseReleaseEvent** (QMouseEvent *event, bool panning)
- bool **markupMouseMoveEvent** (QMouseEvent *event, bool panning)
- void **markupResize** (QSize newSize, double scale)
- virtual void **markupChange** (QImage &markups, QVector< QRect > &changedAreas)=0
- virtual void **markupAction** (markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)=0

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.20 loginWidget Class Reference

Public Member Functions

- **loginWidget** (QELogin *ownerIn)
- userLevelTypes::userLevels **getUserType** ()
- QString **getPassword** ()

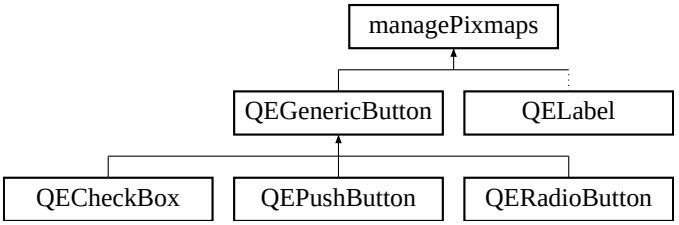
- void **clearPassword** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.21 managePixmaps Class Reference

Inheritance diagram for managePixmaps:



Public Member Functions

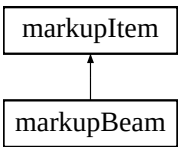
- void **setDataPixmap** (const QPixmap &Pixmap, const unsigned int index)
- QPixmap **getDataPixmap** (const unsigned int index)
- QPixmap **getDataPixmap** (const QString value)

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

- /tmp/epicsqt/trunk/framework/widgets/include/managePixmaps.h
- /tmp/epicsqt/trunk/framework/widgets/src/managePixmaps.cpp

9.22 markupBeam Class Reference

Inheritance diagram for markupBeam:



Public Member Functions

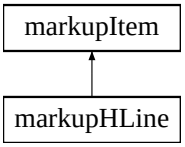
- **markupBeam** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.23 markupHLine Class Reference

Inheritance diagram for markupHLine:



Public Member Functions

- **markupHLine** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void drawMarkup (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)

- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

9.23.1 Member Function Documentation

9.23.1.1 void markupHLine::drawMarkup (QPainter & *p*) [virtual]

!! draw the handle in the middle of the existing view, not the entire image

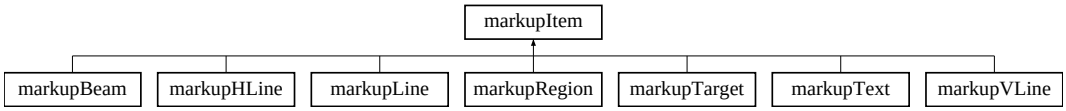
Implements markupItem.

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.24 markupItem Class Reference

Inheritance diagram for markupItem:



Public Types

- enum **markupHandles** {

  MARKUP_HANDLE_NONE, **MARKUP_HANDLE_START**, **MARKUP_HANDLE_-END**, **MARKUP_HANDLE_CENTER**,

  MARKUP_HANDLE_TL, **MARKUP_HANDLE_TR**, **MARKUP_HANDLE_BL**, **MARKUP_-HANDLE_BR**,

  MARKUP_HANDLE_T, **MARKUP_HANDLE_B**, **MARKUP_HANDLE_L**, **MARKUP_-HANDLE_R** }

Public Member Functions

- void **erase** ()
- void **drawMarkupIn** ()

- void **drawMarkupOut** ()
- void **setColor** (QColor colorIn)
- void **scale** (const double xScale, const double yScale, const double zoomScale)
- virtual QPoint **origin** ()=0
- virtual void **moveTo** (const QPoint pos)=0
- virtual void **startDrawing** (const QPoint pos)=0
- virtual bool **isOver** (const QPoint point, QCursor ∗cursor)=0
- virtual QCursor **cursorForHandle** (const markupItem::markupHandles handle)=0

- virtual QPoint **getPoint1** ()=0
- virtual QPoint **getPoint2** ()=0
- virtual unsigned int **getThickness** ()=0
- virtual void **setThickness** (const unsigned int thicknessIn)=0
- virtual QCursor **defaultCursor** ()=0
- virtual void **nonInteractiveUpdate** (QRect)

Public Attributes

- QRect **area**
- bool **visible**
- bool **interactive**
- bool **reportOnMove**
- QColor **color**

Protected Types

- enum **isOverOptions** { **OVER_LINE**, **OVER_BORDER**, **OVER_AREA** }
- enum **legendJustification** { **ABOVE_RIGHT**, **BELOW_LEFT**, **BELOW_RIGHT** }

Protected Member Functions

- **markupItem** (imageMarkup ∗ownerIn, const isOverOptions over, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- virtual void **setArea** ()=0
- virtual void **drawMarkup** (QPainter &p)=0
- bool **pointIsNear** (QPoint p1, QPoint p)
- QColor **getColor** ()
- const QString **getLegend** ()
- const QSize **getLegendSize** ()
- void **addLegendArea** ()
- const QPoint **setLegendPos** (QPoint pos, legendJustification just)
- const QPoint **getLegendPos** ()
- void **drawLegend** (QPainter &p, QPoint pos, legendJustification just)
- QPoint **limitPointToImage** (const QPoint pos)

Protected Attributes

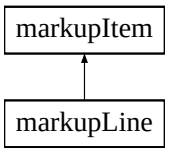
- markupHandles **activeHandle**
- isOverOptions **isOverType**
- bool **highlighted**
- int **highlightMargin**
- imageMarkup * **owner**

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.25 markupLine Class Reference

Inheritance diagram for markupLine:



Public Member Functions

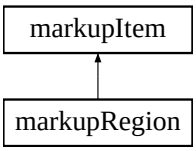
- **markupLine** (imageMarkup *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.26 markupRegion Class Reference

Inheritance diagram for markupRegion:



Public Member Functions

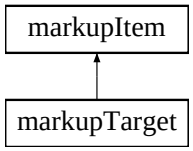
- **markupRegion** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QRect)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.27 markupTarget Class Reference

Inheritance diagram for markupTarget:



Public Member Functions

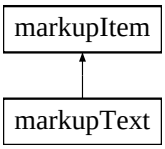
- **markupTarget** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.28 markupText Class Reference

Inheritance diagram for markupText:



Public Member Functions

- **markupText** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **setText** (QString textIn, bool draw)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()

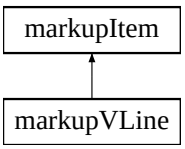
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.29 markupVLine Class Reference

Inheritance diagram for markupVLine:



Public Member Functions

- **markupVLine** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void drawMarkup (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

9.29.1 Member Function Documentation

9.29.1.1 void markupVLine::drawMarkup (QPainter & *p*) [virtual]

!! draw the handle in the middle of the existing view, not the entire image

Implements markupItem.

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.30 message_types Class Reference

Public Member Functions

- **message_types** (message_severities severityIn, message_kind_sets kind_setIn=MESSAGE_KIND_STANDARD)
- QString getSeverityName ()

  Function to provide string name for each message type severity.

Public Attributes

- message_severities **severity**
- message_kind_sets **kind_set**

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

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.31 QEStripChartToolBar::OwnWidgets Class Reference

Public Member Functions

- **OwnWidgets** (QEStripChartToolBar *parent)

Public Attributes

- QPushButton * **pushButtons** [NUMBER_OF_BUTTONS]
- QLabel * **timeStatus**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.32 QEPvProperties::OwnWidgets Class Reference

Public Member Functions

- **OwnWidgets** (QEPvProperties ∗parent)

Public Attributes

- QFrame ∗ **topFrame**
- QLabel ∗ **label1**
- QLabel ∗ **label2**
- QLabel ∗ **label3**
- QLabel ∗ **label4**
- QLabel ∗ **label5**
- QLabel ∗ **label6**
- QComboBox ∗ **box**
- QELabel ∗ **valueLabel**
- QLabel ∗ **hostName**
- QLabel ∗ **fieldType**
- QLabel ∗ **timeStamp**
- QLabel ∗ **indexInfo**
- QVBoxLayout ∗ **topFrameVlayout**
- QHBoxLayout ∗ **hlayouts** [6]
- QTableWidget ∗ **table**
- QMenu ∗ **tableContextMenu**
- QFrame ∗ **enumerationFrame**
- QLabelList **enumerationLabelList**
- QScrollArea ∗ **enumerationScroll**
- QEResizeableFrame ∗ **enumerationResize**
- QVBoxLayout ∗ **vlayout**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.33 PeriodicDialog Class Reference

Public Member Functions

- **PeriodicDialog** (QWidget ∗parent=0)
- QString **getElement** ()
- void **setElement** (QString elementIn, QList< bool > &enabledList, QList< QString > &elementList)

Protected Member Functions

- void **changeEvent** (QEvent ∗e)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicDialog.cpp

9.34 PeriodicElementSetupForm Class Reference

Public Member Functions

- **PeriodicElementSetupForm** (QWidget ∗parent=0)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicElementSetupForm.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicElementSetupForm.cpp

9.35 PeriodicSetupDialog Class Reference

Public Member Functions

- **PeriodicSetupDialog** (QWidget ∗parent=0)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicSetupDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicSetupDialog.cpp

9.36 PersistanceManager Class Reference

Public Member Functions

- QObject ∗ **getSaveRestoreObject** ()
- void **save** (const QString fileName, const QString rootName, const QString configName)
- void **restore** (const QString fileName, const QString rootName, const QString configName)
- PMElement **addNamedConfiguration** (QString name)
- PMElement **getNamedConfiguration** (QString name)
- QStringList **getConfigNames** (QString fileName, QString rootName)
- void **deleteConfigs** (QString fileName, QString rootName, QStringList names)

Friends

- class PMElement

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.37 PMContext Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h

9.38 PMElement Class Reference

Public Member Functions

- **PMElement** (PersistanceManager ∗ownerIn, QDomElement elementIn)
- PMElement **addElement** (QString name)
- void **addValue** (QString name, bool value)
- void **addValue** (QString name, int value)
- void **addValue** (QString name, double value)
- void **addValue** (QString name, QString value)
- void **addAttribute** (QString name, bool value)
- void **addAttribute** (QString name, int value)
- void **addAttribute** (QString name, double value)
- void **addAttribute** (QString name, QString value)
- PMElement **getElement** (QString name)
- PMElement **getElement** (QString name, int i)
- PMElement **getElement** (QString name, QString attrName, QString attrValue)
- PMElement **getElement** (QString name, QString attrName, int attrValue)
- PMElementList **getElementList** (QString name)
- bool **getValue** (QString name, bool &val)
- bool **getValue** (QString name, int &val)
- bool **getValue** (QString name, double &val)
- bool **getValue** (QString name, QString &val)
- bool **getAttribute** (QString name, bool &val)
- bool **getAttribute** (QString name, int &val)
- bool **getAttribute** (QString name, double &val)
- bool **getAttribute** (QString name, QString &val)
- bool **isNull** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.39 PMElementList Class Reference

Public Member Functions

- **PMElementList** (PersistanceManager ∗ownerIn, QDomNodeList elementListIn)
- PMElement getElement (int i)
- int **count** ()

9.39.1 Member Function Documentation

9.39.1.1 PMElement PMElementList::getElement (int *i*)

!! check range of i

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.40 QEStripChart::PrivateData Class Reference

Public Member Functions

- **PrivateData** (QEStripChart ∗chartIn)
- QEStripChartItem ∗ **getItem** (unsigned int slot)
- QwtPlotCurve ∗ **allocateCurve** ()
- void **calcDisplayMinMax** ()
- void **plotData** ()
- void **setReadOut** (const QString &text)
- void **setNormalBackground** (bool state)
- void **chartContextMenuRequested** (const QPoint &pos)
- void **nullContextMenuRequested** (const QPoint &pos)
- void **chartContextMenuTriggered** (QAction ∗action)
- void **itemContextMenuRequested** (const unsigned int slot, const QPoint &pos)
- void **itemContextMenuSelected** (const unsigned int, const QEStripChartNames::ContextMenuOptions option)
- void **pushState** ()
- void **prevState** ()
- void **nextState** ()
- void **captureState** (ChartState &chartState)
- void **applyState** (const ChartState &chartState)

Public Attributes

- QEStripChartNames::ChartYRanges **chartYScale**
- QEStripChartNames::YScaleModes **yScaleMode**
- QEStripChartNames::ChartTimeModes **chartTimeMode**
- double **timeScale**
- QString **timeUnits**

Protected Member Functions

- bool **eventFilter** (QObject ∗obj, QEvent ∗event)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.41 QEStripChartItem::PrivateData Class Reference

Public Attributes

- QEStripChart ∗ **chart**
- QLabel ∗ **pvName**
- QELabel ∗ **caLabel**
- QColorDialog ∗ **colourDialog**
- qcaobject::QCaObject ∗ **previousQcaItem**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.42 profilePlot Class Reference

Public Types

- enum **plotDirections** { **PROFILEPLOT_LR**, **PROFILEPLOT_RL**, **PROFILEPLOT_-TB**, **PROFILEPLOT_BT** }

Public Member Functions

- **profilePlot** (plotDirections plotDirectionIn)
- void **setProfile** (QVector< QPointF > ∗profile, double minX, double maxX, double minY, double maxY, QString title, QPoint start, QPoint end, unsigned int thicknessIn)

- void **clearProfile** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/profilePlot.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/profilePlot.cpp

9.43 PublishedProfile Class Reference

Public Attributes

- QObject * **guiLaunchConsumer**
- QStringList **pathList**
- QString **parentPath**
- QList< QString > **macroSubstitutions**
- unsigned int **messageFormId**
- QList< WidgetRef > **containedWidgets**
- userLevelSignal **userSignal**
- QString **userLevelPassword**
- QString **scientistLevelPassword**
- QString **engineerLevelPassword**
- bool **profileDefined**
- PersistanceManager **persistanceManager**
- bool **dontActivateYet**
- bool **userLevelPasswordsSet**

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.44 PushButtonSpecifications Struct Reference

Public Attributes

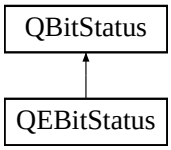
- int **gap**
- int **width**
- bool **isIcon**
- const QString **captionOrIcon**
- const QString **toolTip**
- const char * **member**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.45 QBitStatus Class Reference

Inheritance diagram for QBitStatus:



Public Types

- enum **Orientations** { **LSB_On_Right**, **LSB_On_Bottom**, **LSB_On_Left**, **LSB_-On_Top** }
- enum **Shapes** { **Rectangle**, **Circle** }

Public Slots

- void **setValue** (const int value)

Public Member Functions

- **QBitStatus** (QWidget ∗parent=0)
- virtual QSize **sizeHint** () const
- void **setBorderColour** (const QColor value)
- QColor **getBorderColour** ()
- void **setOnColour** (const QColor value)
- QColor **getOnColour** ()
- void **setOffColour** (const QColor value)
- QColor **getOffColour** ()
- void **setInvalidColour** (const QColor value)
- QColor **getInvalidColour** ()
- void **setClearColour** (const QColor value)
- QColor **getClearColour** ()
- void **setDrawBorder** (const bool value)
- bool **getDrawBorder** ()
- void **setNumberOfBits** (const int value)
- int **getNumberOfBits** ()
- void **setGap** (const int value)
- int **getGap** ()
- void **setShift** (const int value)
- int **getShift** ()
- void **setOnClearMask** (const QString value)
- QString **getOnClearMask** ()

- void **setOffClearMask** (const QString value)
- QString **getOffClearMask** ()
- void **setReversePolarityMask** (const QString value)
- QString **getReversePolarityMask** ()
- void **setIsValid** (const bool value)
- bool **getIsValid** ()
- void **setOrientation** (const enum Orientations value)
- enum Orientations **getOrientation** ()
- void **setShape** (const enum Shapes value)
- enum Shapes **getShape** ()
- int **getValue** ()

Properties

- int **value**
- int **numberOfBits**
- int **shift**
- Orientations **Orientation**
- Shapes **shape**
- int **gap**
- QString **reversePolarityMask**
- QString **onClearMask**
- QString **offClearMask**
- QColor **boarderColour**
- QColor **invalidColour**
- QColor **onColour**
- QColor **offColour**
- QColor **clearColour**
- bool **drawBorder**
- bool **isValid**

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

- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QBitStatus.h
- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QBitStatus.cpp

9.46 QCaAlarmInfo Class Reference

Public Member Functions

- **QCaAlarmInfo** (unsigned short statusIn, unsigned short severityIn)
- QString **statusName** ()
- QString **severityName** ()
- bool **isInAlarm** ()
- bool **isMinor** ()

- bool **isMajor** ()
- bool **isInvalid** ()
- QString **style** ()
- QString **getColorName** ()
- QCAALARMINFO_SEVERITY **getSeverity** ()

Static Public Member Functions

- static QCAALARMINFO_SEVERITY **getInvalidSeverity** ()

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

- /tmp/epicsqt/trunk/framework/data/include/QCaAlarmInfo.h
- /tmp/epicsqt/trunk/framework/data/src/QCaAlarmInfo.cpp

9.47 QCaConnectionInfo Class Reference

Public Member Functions

- **QCaConnectionInfo** (unsigned short channelStateIn, unsigned short linkStateIn)

- bool **isChannelConnected** ()
- bool **isLinkUp** ()

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

- /tmp/epicsqt/trunk/framework/data/include/QCaConnectionInfo.h
- /tmp/epicsqt/trunk/framework/data/src/QCaConnectionInfo.cpp

9.48 QCaDataPoint Class Reference

Public Member Functions

- bool **isDisplayable** ()

Public Attributes

- double **value**
- QCaDateTime **datetime**
- QCaAlarmInfo **alarm**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDataPoint.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDataPoint.cpp

9.49 QCaDataPointList Class Reference

Public Member Functions

- void **resample** (const QCaDataPointList &source, const double interval, const QCaDateTime &endTime)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDataPoint.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDataPoint.cpp

9.50 QCaDateTime Class Reference

Public Member Functions

- **QCaDateTime** (QDateTime dt)
- QCaDateTime & **operator=** (const QCaDateTime &other)
- **QCaDateTime** (unsigned long seconds, unsigned long nanoseconds)
- QString **text** ()
- double floating (const QDateTime &base) const
- unsigned long getSeconds () const

  Recover original EPICS time constructor parameters.
- unsigned long **getNanoSeconds** () const

9.50.1 Member Function Documentation

9.50.1.1 double QCaDateTime::floating (const QDateTime & *base*) const

Duration in seconds from base time to this time. Note: this is the opposite sense to the parent QDateTime daysTo, secsTo and msecsTo functions.

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDateTime.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDateTime.cpp

9.51 QCaEventFilter Class Reference

Public Member Functions

- void **addFilter** (QObject ∗objectIn)
- void **deleteFilter** (QObject ∗objectIn)
- bool **eventFilter** (QObject ∗watched, QEvent ∗e)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaEventFilter.h
- /tmp/epicsqt/trunk/framework/data/src/QCaEventFilter.cpp

9.52 QCaEventItem Class Reference

Public Member Functions

- **QCaEventItem** (QCaEventUpdate *newEvent)

Public Attributes

- QCaEventUpdate * **event**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaEventUpdate.h

9.53 QCaEventUpdate Class Reference

Public Member Functions

- **QCaEventUpdate** (qcaobject::QCaObject *emitterObjectIn, long newReason, void *newDataPtr)

Public Attributes

- bool **acceptThisEvent**
- qcaobject::QCaObject * **emitterObject**
- long **reason**
- void * **dataPtr**

Static Public Attributes

- static QEvent::Type **EVENT_UPDATE_TYPE** = QEvent::User

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

- /tmp/epicsqt/trunk/framework/data/include/QCaEventUpdate.h
- /tmp/epicsqt/trunk/framework/data/src/QCaEventUpdate.cpp

9.54 QCaInstalledFiltersListItem Class Reference

Public Member Functions

- **QCaInstalledFiltersListItem** (QObject ∗eventObjectIn)

Public Attributes

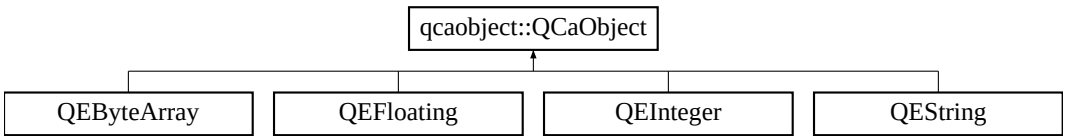
- QObject ∗ **eventObject**
- long **referenceCount**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaEventFilter.h

9.55 qcaobject::QCaObject Class Reference

Inheritance diagram for qcaobject::QCaObject:



Public Slots

- bool **writeData** (const QVariant &value)
- void **resendLastData** ()

Signals

- void **dataChanged** (const QVariant &value, QCaAlarmInfo &alarmInfo, QCaDate-Time &timeStamp)
- void **dataChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp)
- void **connectionChanged** (QCaConnectionInfo &connectionInfo)

Public Member Functions

- **QCaObject** (const QString &recordName, QObject ∗eventObject, unsigned char signalsToSendIn=SIG_VARIANT)

- **QCaObject** (const QString &recordName, QObject ∗eventObject, UserMessage ∗userMessageIn, unsigned char signalsToSendIn=SIG_VARIANT)
- bool **subscribe** ()
- bool **singleShotRead** ()
- bool **dataTypeKnown** ()
- bool **createChannel** ()
- void **deleteChannel** ()
- bool **createSubscription** ()
- bool **getChannel** ()
- bool **putChannel** ()
- bool **isChannelConnected** ()
- void **startConnectionTimer** ()
- void **stopConnectionTimer** ()
- void **setUserMessage** (UserMessage ∗userMessageIn)
- void **enableWriteCallbacks** (bool enable)
- bool **isWriteCallbacksEnabled** ()
- QString **getRecordName** ()
- QString **getEgu** ()
- QStringList **getEnumerations** ()
- unsigned int **getPrecision** ()
- QCaAlarmInfo **getAlarmInfo** ()
- QCaDateTime **getDateTime** ()
- double **getDisplayLimitUpper** ()
- double **getDisplayLimitLower** ()
- double **getAlarmLimitUpper** ()
- double **getAlarmLimitLower** ()
- double **getWarningLimitUpper** ()
- double **getWarningLimitLower** ()
- double **getControlLimitUpper** ()
- double **getControlLimitLower** ()
- generic::generic_types **getDataType** ()
- QString **getHostName** ()
- QString **getFieldType** ()
- unsigned long **getElementCount** ()
- void **getLastData** (bool &isDefined, QVariant &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp)

Static Public Member Functions

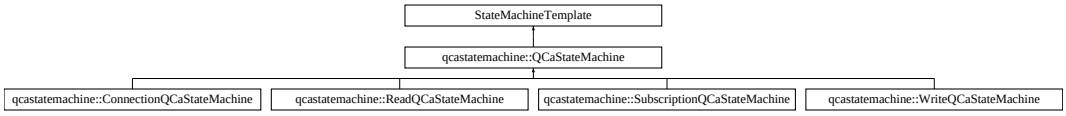
- static void **processEventStatic** (QCaEventUpdate ∗dataUpdateEvent)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaObject.h
- /tmp/epicsqt/trunk/framework/data/src/QCaObject.cpp

9.56 qcastatemachine::QCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::QCaStateMachine:



Public Member Functions

- **QCaStateMachine** (void ∗parent)
- virtual bool **process** (int requestedState)=0

Public Attributes

- QMutex **lock**
- bool **pending**
- bool **active**
- bool **expired**
- void ∗ **myWorker**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.57 QCaVariableNamePropertyManager Class Reference

Signals

- void **newVariableNameProperty** (QString variable, QString Substitutions, unsigned int variableIndex)

Public Member Functions

- QString **getVariableNameProperty** ()
- void **setVariableNameProperty** (QString variableNamePropertyIn)
- QString **getSubstitutionsProperty** ()
- void **setSubstitutionsProperty** (QString substitutionsPropertyIn)
- void **setVariableIndex** (unsigned int variableIndexIn)

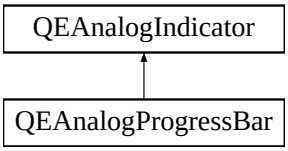
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/data/include/QCaVariableNamePropertyManager.h
- /tmp/epicsqt/trunk/framework/data/src/QCaVariableNamePropertyManager.cpp

9.58 QEAnalogIndicator Class Reference

`#include <QEAnalogIndicator.h>`

Inheritance diagram for QEAnalogIndicator:



Classes

- struct Band
- class BandList

Public Types

- enum Orientations { Left_To_Right, Top_To_Bottom, Right_To_Left, Bottom_To_-Top }
- enum Modes { Bar, Scale, Meter }

Public Slots

- void **setRange** (const double MinimumIn, const double MaximumIn)
- void **setValue** (const double ValueIn)

Public Member Functions

- QEAnalogIndicator (QWidget *parent=0)

  Constructor.
- virtual ~QEAnalogIndicator ()

  Destructor.
- virtual QSize sizeHint () const

  Size hint.
- double getValue ()

  Access function for value property - refer to value property for details.
- void setMinimum (const double value)

  Access function for minimum - refer to minimum property for details.

- double getMinimum ()

  Access function for minimum - refer to minimum property for details.
- void setMaximum (const double value)

  Access function for maximum - refer to maximum property for details.
- double getMaximum ()

  Access function for maximum - refer to maximum property for details.
- void setOrientation (const enum Orientations value)

  Access function for orientation - refer to orientation property for details.
- enum Orientations getOrientation ()

  Access function for orientation - refer to orientation property for details.
- void setMode (const enum Modes value)

  Access function for mode - refer to mode property for details.
- enum Modes getMode ()

  Access function for mode - refer to mode property for details.
- void setCentreAngle (const int value)

  Access function for centreAngle - refer to centreAngle property for details.
- int getCentreAngle ()

  Access function for centreAngle - refer to centreAngle property for details.
- void setSpanAngle (const int value)

  Access function for spanAngle - refer to spanAngle property for details.
- int getSpanAngle ()

  Access function for spanAngle - refer to spanAngle property for details.
- void setMinorInterval (const double value)

  Access function for minorInterval - refer to minorInterval property for details.
- double getMinorInterval ()

  Access function for minorInterval - refer to minorInterval property for details.
- void setMajorInterval (const double value)

  Access function for majorInterval - refer to majorInterval property for details.
- double getMajorInterval ()

  Access function for majorInterval - refer to majorInterval property for details.
- void setLogScaleInterval (const int value)

  Access function for logScaleInterval - refer to logScaleInterval property for details.
- int getLogScaleInterval ()

  Access function for logScaleInterval - refer to logScaleInterval property for details.
- void setBorderColour (const QColor value)

  Access function for borderColour - refer to borderColour property for details.
- QColor getBorderColour ()

  Access function for borderColour - refer to borderColour property for details.
- void setForegroundColour (const QColor value)

  Access function for foregroundColour - refer to foregroundColour property for details.
- QColor getForegroundColour ()

  Access function for foregroundColour - refer to foregroundColour property for details.
- void setBackgroundColour (const QColor value)

Access function for backgroundColour - refer to backgroundColour property for details.

- QColor getBackgroundColour ()

  Access function for backgroundColour - refer to backgroundColour property for details.
- void setFontColour (const QColor value)

  Access function for fontColour - refer to fontColour property for details.
- QColor getFontColour ()

  Access function for fontColour - refer to fontColour property for details.
- void setShowText (const bool value)

  Access function for showText - refer to showText property for details.
- bool getShowText ()

  Access function for showText - refer to showText property for details.
- void setShowScale (const bool value)

  Access function for showScale - refer to showScale property for details.
- bool getShowScale ()

  Access function for showScale - refer to showScale property for details.
- void setLogScale (const bool value)

  Access function for logScale - refer to logScale property for details.
- bool getLogScale ()

  Access function for logScale - refer to logScale property for details.

Protected Member Functions

- virtual QString **getTextImage** ()
- virtual BandList **getBandList** ()

Properties

- double value
- double minimum
- double maximum
- double minorInterval
- double majorInterval
- int logScaleInterval
- bool showText
- bool showScale
- bool logScale
- Modes mode
- Orientations orientation
- int centreAngle
- int spanAngle
- QColor borderColour
- QColor backgroundColour
- QColor foregroundColour
- QColor fontColour

9.58.1 Detailed Description

This class provides a non CA aware graphical analog indicator base class. It supports a number of display modes including Bar, Scale and Meter.

When in Bar mode, it mimics QProgressBar and provides an analog progress bar widget.

9.58.2 Member Enumeration Documentation

9.58.2.1 enum QEAnalogIndicator::Modes

The type of analog indicator used to represent the value

Enumerator:

Bar Bar (solid bar from minimum up to current value)

Scale Scale (diamond marker tracks current value)

Meter Meter (Needle moving across an arc scale)

9.58.2.2 enum QEAnalogIndicator::Orientations

The orientation of Bar and Scale indicators

Enumerator:

Left_To_Right Left to right.

Top_To_Bottom Top to bottom.

Right_To_Left Right to left.

Bottom_To_Top Bottom to top.

9.58.3 Property Documentation

9.58.3.1 QColor QEAnalogIndicator::backgroundColour [read, write]

Background colour

9.58.3.2 QColor QEAnalogIndicator::borderColour [read, write]

Border colour

9.58.3.3 int QEAnalogIndicator::centreAngle [read, write]

The angle in degreed of the line that Meter indicators are centered around. Zero represents a vertical centerline and angles increment clockwise.

9.58.3.4 QColor QEAnalogIndicator::fontColour [read, write]

Font colour

9.58.3.5 QColor QEAnalogIndicator::foregroundColour [read, write]

Foreground colour

9.58.3.6 bool QEAnalogIndicator::logScale [read, write]

If set, use a logarithmic scale. If clear, use a linear scale

9.58.3.7 int QEAnalogIndicator::logScaleInterval [read, write]

Log scale interval.

9.58.3.8 double QEAnalogIndicator::majorInterval [read, write]

Minor scale interval. Only applies for linear scale (not log scale)

9.58.3.9 double QEAnalogIndicator::maximum [read, write]

Maximum indicated value.

9.58.3.10 double QEAnalogIndicator::minimum [read, write]

Minimum indicated value.

9.58.3.11 double QEAnalogIndicator::minorInterval [read, write]

Minor scale interval. Only applies for linear scale (not log scale)

9.58.3.12 Modes QEAnalogIndicator::mode [read, write]

Selects what type of indicator is used (refer to Modes)

9.58.3.13 Orientations QEAnalogIndicator::orientation [read, write]

The orientation of Bar and Scale indicators (refer to Orientations)

9.58.3.14 bool QEAnalogIndicator::showScale [read, write]

If set, show the scale

9.58.3.15 bool QEAnalogIndicator::showText [read, write]

If set, show textual representation of value on the indicator

9.58.3.16 int QEAnalogIndicator::spanAngle [read, write]

The span of the Meter scale arc in degrees Typical meters are 180 deg and 270 deg

9.58.3.17 double QEAnalogIndicator::value [read, write]

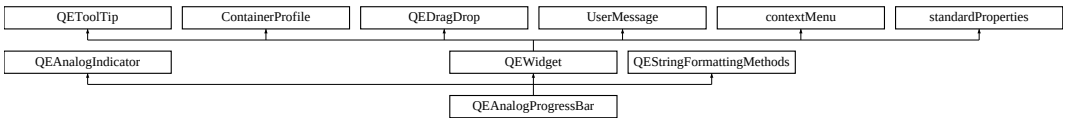
Current indicated value.

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h
- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.cpp

9.59 QEAnalogProgressBar Class Reference

Inheritance diagram for QEAnalogProgressBar:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum **AlarmSeverityDisplayModes** { **foreground**, **background** }
- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInteger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_AUTOMATIC }

- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEStringFormatting::ASCII, Index = QEStringFormatting::INDEX }

Signals

- void dbValueChanged (const double &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.
- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.
- QEAnalogProgressBar (QWidget *parent=0)
- QEAnalogProgressBar (const QString &variableName, QWidget *parent=0)
- virtual ~QEAnalogProgressBar ()

  Destruction.
- void setUseDbDisplayLimits (bool useDbDisplayLimitsIn)

  Access function for useDbDisplayLimits property - refer to useDbDisplayLimits property for details.
- bool getUseDbDisplayLimits ()

Access function for useDbDisplayLimits property - refer to useDbDisplayLimits property for details.

- void setAlarmSeverityDisplayMode (AlarmSeverityDisplayModes value)

  Access function for #AlarmSeverityDisplayModes property - refer to #AlarmSeverityDisplayModes property for details.

- AlarmSeverityDisplayModes getAlarmSeverityDisplayMode ()

  Access function for #AlarmSeverityDisplayModes property - refer to #AlarmSeverityDisplayModes property for details.

Protected Member Functions

- QString **getTextImage** ()
- BandList **getBandList** ()
- void **establishConnection** (unsigned int variableIndex)
- void **stringFormattingChange** ()
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- AlarmSeverityDisplayModes alarmSeverityDisplayMode
- bool useDbDisplayLimits
- int precision
- bool useDbPrecision

- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction

9.59.1 Member Enumeration Documentation

9.59.1.1 enum QEAnalogProgressBar::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.59.1.2 enum QEAnalogProgressBar::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.59.1.3 enum QEAnalogProgressBar::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.59.1.4 enum QEAnalogProgressBar::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.59.2 Constructor & Destructor Documentation

9.59.2.1 QEAnalogProgressBar::QEAnalogProgressBar (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.59.2.2 QEAnalogProgressBar::QEAnalogProgressBar (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.59.3 Member Function Documentation

9.59.3.1 void QEAnalogProgressBar::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.59.4 Property Documentation

9.59.4.1 bool QEAnalogProgressBar::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.59.4.2 AlarmSeverityDisplayModes QEAnalogProgressBar::alarmSeverityDisplayMode [read, write]

Visualise the EPICS alarm severity

9.59.4.3 bool QEAnalogProgressBar::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.59.4.4 ArrayActions QEAnalogProgressBar::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.59.4.5 bool QEAnalogProgressBar::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.59.4.6 Formats QEAnalogProgressBar::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.59.4.7 unsigned QEAnalogProgressBar::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.59.4.8 bool QEAnalogProgressBar::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.59.4.9 QString QEAnalogProgressBar::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.59.4.10 Notations QEAnalogProgressBar::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.59.4.11 int QEAnalogProgressBar::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.59.4.12 bool QEAnalogProgressBar::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.59.4.13 bool QEAnalogProgressBar::useDbDisplayLimits [read, write]

Use the EPICS database display limits

9.59.4.14 bool QEAnalogProgressBar::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.59.4.15 UserLevels QEAnalogProgressBar::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.59.4.16 QString QEAnalogProgressBar::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.59.4.17 QString QEAnalogProgressBar::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.59.4.18 QString QEAnalogProgressBar::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example,

'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.59.4.19 UserLevels QEAnalogProgressBar::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.59.4.20 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

9.59.4.21 bool QEAnalogProgressBar::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.59.4.22 QString QEAnalogProgressBar::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.59.4.23 bool QEAnalogProgressBar::visible [read, write]

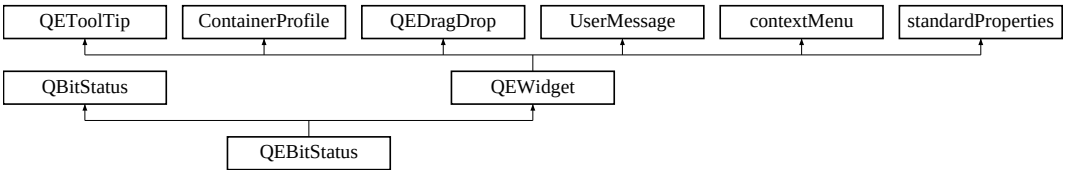
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.h
- /tmp/epicsqt/trunk/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.cpp

9.60 QEBitStatus Class Reference

Inheritance diagram for QEBitStatus:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void dbValueChanged (const long &out)

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- **QEBitStatus** (QWidget *parent=0)
- **QEBitStatus** (const QString &variableName, QWidget *parent=0)
- void setVariableNameAndSubstitutions (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)

- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()

Protected Attributes

- QEIntegerFormatting **integerFormatting**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.60.1 Member Enumeration Documentation

9.60.1.1 enum QEBitStatus::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.60.2 Member Function Documentation

9.60.2.1 void QEBitStatus::dbValueChanged (const long & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.60.2.2 void QEBitStatus::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) [virtual]

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

Reimplemented from QEWidget.

9.60.3 Property Documentation

9.60.3.1 bool QEBitStatus::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.60.3.2 bool QEBitStatus::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.60.3.3 unsigned QEBitStatus::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.60.3.4 UserLevels QEBitStatus::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.60.3.5 QString QEBitStatus::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example,

'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.60.3.6 QString QEBitStatus::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.60.3.7 QString QEBitStatus::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.60.3.8 UserLevels QEBitStatus::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.60.3.9 QString QEBitStatus::variable [read, write]

EPICS variable name (CA PV)

9.60.3.10 bool QEBitStatus::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.60.3.11 QString QEBitStatus::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.60.3.12 bool QEBitStatus::visible [read, write]

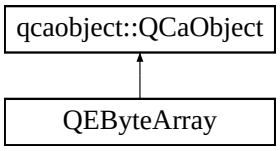
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QEBitStatus.h
- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QEBitStatus.cpp

9.61 QEByteArray Class Reference

Inheritance diagram for QEByteArray:



Public Slots

- void **writeByteArray** (const QByteArray &data)

Signals

- void **byteArrayConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **byteArrayChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEByteArray** (QString recordName, QObject *eventObject, unsigned int variableIndexIn)

- **QEByteArray** (QString recordName, QObject ∗eventObject, unsigned int variableIndexIn, UserMessage ∗userMessageIn)

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

- /tmp/epicsqt/trunk/framework/data/include/QEByteArray.h
- /tmp/epicsqt/trunk/framework/data/src/QEByteArray.cpp

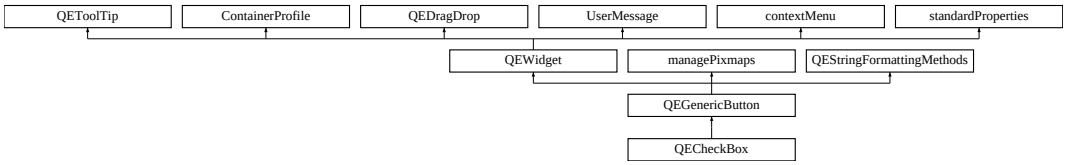
9.62 QEChartStateLists Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.63 QECheckBox Class Reference

Inheritance diagram for QECheckBox:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInteger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEStringFormatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QEGenericButton::UPDATE_TEXT, Icon = QEGenericButton::UPDATE_ICON, TextAndIcon = QEGenericButton::UPDATE_TEXT_AND_ICON, State = QEGenericButton::UPDATE_STATE }

User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.

- enum CreationOptionNames { Open = QEForm::CREATION_OPTION_OPEN, NewTab = QEForm::CREATION_OPTION_NEW_TAB, NewWindow = QEForm::CREATION_OPTION_NEW_WINDOW }

  Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Public Slots

- void launchGui (QString guiName, QEForm::creationOptions creationOption)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.
- void newGui (QString guiName, QEForm::creationOptions creationOption)

  Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.
- void pressed (int value)
- void released (int value)
- void clicked (int value)

Public Member Functions

- QECheckBox (QWidget *parent=0)
- QECheckBox (const QString &variableName, QWidget *parent=0)
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.

- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- Qt::Alignment alignment
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7
- QString password
- bool confirmAction

- bool writeOnPress
- bool writeOnRelease
- bool writeOnClick
- QString pressText
- QString releaseText
- QString clickText
- QString clickCheckedText
- QString labelText
- QString program
- QStringList arguments
- QString guiFile
- CreationOptionNames creationOption
- QString prioritySubstitutions

9.63.1 Member Enumeration Documentation

9.63.1.1 enum QECheckBox::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.63.1.2 enum QECheckBox::CreationOptionNames

Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Enumerator:

Open Replace the current GUI with the new GUI.

NewTab Open new GUI in a new tab.

NewWindow Open new GUI in a new window.

9.63.1.3 enum QECheckBox::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.63.1.4 enum QECheckBox::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.63.1.5 enum QECheckBox::UpdateOptions

User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.

Enumerator:

Text Data updates will update the button text.

Icon Data updates will update the button icon.

TextAndIcon Data updates will update the button text and icon.

State Data updates will update the button state (checked or unchecked)

9.63.1.6 enum QECheckBox::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.63.2 Constructor & Destructor Documentation

9.63.2.1 QECheckBox::QECheckBox (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.63.2.2 QECheckBox::QECheckBox (const QString & *variableName,* QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.63.3 Member Function Documentation

9.63.3.1 void QECheckBox::clicked (int *value*) [signal]

Button has been Clicked. The value emitted is the integer interpretation of the clickText property (or the clickCheckedText property if the button was checked)

9.63.3.2 void QECheckBox::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.63.3.3 void QECheckBox::launchGui (QString *guiName,* QEForm::creationOptions *creationOption*) [inline, slot]

Default slot used to create a new GUI if there is no slot indicated in the ContainerProfile class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not specify a slot to use for creating new windows (through the ContainerProfile class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the ContainerProfile class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from QEGenericButton.

9.63.3.4 void QECheckBox::pressed (int *value*) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the pressText property

9.63.3.5 void QECheckBox::released (int *value*) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

9.63.4 Property Documentation

9.63.4.1 bool QECheckBox::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.63.4.2 Qt::Alignment QECheckBox::alignment [read, write]

Set the buttons text alignment. Left justification is particularly useful when displaying quickly changing numeric data updates.

9.63.4.3 bool QECheckBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.63.4.4 QStringList QECheckBox::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from QEGenericButton.

9.63.4.5 ArrayActions QECheckBox::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.63.4.6 QString QECheckBox::clickCheckedText [read, write]

Text used to compare with text written or read to determine if push button should be marked as checked. Note, must be an exact match following formatting of data updates. When writing values, the 'pressText', 'ReleaseText', or 'clickedtext' must match this property to cause the button to be checked when the write occurs.

Good example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is 'On'. In this example, the push button will be checked when a data update occurs with a value of 1 or when the button is clicked.

Bad example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is '1'. In this example, the push button will be checked when a data update occurs with a value of 1 but, although a valid value will be written when clicked, the button will not be checked when clicked as '1' is not the same as 'On'.

Reimplemented from QEGenericButton.

9.63.4.7 QString QECheckBox::clickText [read, write]

Value written when user clicks button if 'writeOnClick' property is true

Reimplemented from QEGenericButton.

9.63.4.8 bool QECheckBox::confirmAction [read, write]

If true, a dialog will be presented asking the user to confirm if the button action should be carried out

9.63.4.9 CreationOptionNames QECheckBox::creationOption [read, write]

Creation options when opening a new GUI. Open a new window, open a new tab, or replace the current window. the creation option is supplied when the button generates a newGui signal. Application code connected to this signal should honour this request if possible. When used within the QEGui application, the QEGui application creates a new window, new tab, or replaces the current window as appropriate.

Reimplemented from QEGenericButton.

9.63.4.10 bool QECheckBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.63.4.11 Formats QECheckBox::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.63.4.12 QString QECheckBox::guiFile [read, write]

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QEform in which the QEPushButton is located, relative to the any path in the path list published in the ContainerProfile class, or relative to the current path. See QEWidget::openQEFile() in QEWidget.cpp for details.

9.63.4.13 unsigned QECheckBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.63.4.14 QString QECheckBox::labelText [read, write]

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! For example, a button in a sub form may have a 'labelText' property of 'Turn Pump On'. When the sub form is used twice in a main form with substitutions PUMPNUM=1 and PUMPNUM=2 respectively, the two identical buttons in the sub forms will have the labels 'Turn Pump 1 On' and 'Turn Pump 2 On' respectively.

Reimplemented from QEGenericButton.

9.63.4.15 bool QECheckBox::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.63.4.16 QString QECheckBox::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.63.4.17 Notations QECheckBox::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.63.4.18 QString QECheckBox::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.63.4.19 QPixmap QECheckBox::pixmap0 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 0

9.63.4.20 QPixmap QECheckBox::pixmap1 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 1

9.63.4.21 QPixmap QECheckBox::pixmap2 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 2

9.63.4.22 QPixmap QECheckBox::pixmap3 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 3

9.63.4.23 QPixmap QECheckBox::pixmap4 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 4

9.63.4.24 QPixmap QECheckBox::pixmap5 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 5

9.63.4.25 QPixmap QECheckBox::pixmap6 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 6

9.63.4.26 QPixmap QECheckBox::pixmap7 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 7

9.63.4.27 int QECheckBox::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.63.4.28 QString QECheckBox::pressText [read, write]

Value written when user presses button if 'writeOnPress' property is true

Reimplemented from QEGenericButton.

9.63.4.29 QString QECheckBox::prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly usefull when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substitttions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from QEGenericButton.

9.63.4.30 QString QECheckBox::program [read, write]

Program to run when the button is clicked. No attempt to run a program is made if this property is empty. Example: firefox

Reimplemented from QEGenericButton.

9.63.4.31 QString QECheckBox::releaseText [read, write]

Value written when user releases button if 'writeOnRelease' property is true

Reimplemented from QEGenericButton.

9.63.4.32 bool QECheckBox::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.63.4.33 bool QECheckBox::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.63.4.34 UpdateOptions QECheckBox::updateOption [read, write]

Update options (text, pixmap, both, or state (checked or unchecked)

Reimplemented from QEGenericButton.

9.63.4.35 bool QECheckBox::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.63.4.36 UserLevels QECheckBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.63.4.37 QString QECheckBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.4.38 QString QECheckBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.4.39 QString QECheckBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.4.40 UserLevels QECheckBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.63.4.41 QString QECheckBox::variable [read, write]

EPICS variable name (CA PV)

9.63.4.42 bool QECheckBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.63.4.43 QString QECheckBox::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.63.4.44 bool QECheckBox::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.63.4.45 bool QECheckBox::writeOnClick [read, write]

If true, the 'clickText' property is written when the button is clicked. Default is true

Reimplemented from QEGenericButton.

9.63.4.46 bool QECheckBox::writeOnPress [read, write]

If true, the 'pressText' property is written when the button is pressed. Default is false

Reimplemented from QEGenericButton.

9.63.4.47 bool QECheckBox::writeOnRelease [read, write]

If true, the 'releaseText' property is written when the button is released. Default is false

Reimplemented from QEGenericButton.

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBox.cpp

9.64 QECheckBoxManager Class Reference

Public Member Functions

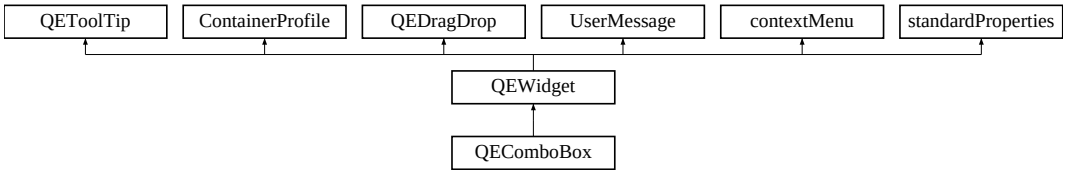
- **QECheckBoxManager** (QObject ∗parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **createWidget** (QWidget ∗parent)
- void **initialize** (QDesignerFormEditorInterface ∗core)

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBoxManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBoxManager.cpp

9.65 QEComboBox Class Reference

Inheritance diagram for QEComboBox:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void dbValueChanged (const qlonglong &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.

Public Member Functions

- **QEComboBox** (QWidget ∗parent=0)
- **QEComboBox** (const QString &variableName, QWidget ∗parent=0)
- void **setWriteOnChange** (bool writeOnChangeIn)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUseDbEnumerations** (bool useDbEnumerations)
- bool **getUseDbEnumerations** ()
- void **setLocalEnumerations** (const QString &localEnumerations)
- QString **getLocalEnumerations** ()
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEIntegerFormatting **integerFormatting**
- QELocalEnumeration **localEnumerations**
- bool useDbEnumerations
- bool writeOnChange

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- QString localEnumeration

9.65.1 Member Enumeration Documentation

9.65.1.1 enum QEComboBox::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.65.2 Member Function Documentation

9.65.2.1 void QEComboBox::dbValueChanged (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.65.3 Member Data Documentation

9.65.3.1 bool QEComboBox::useDbEnumerations [read, write, protected]

Use database enumerations - defaults to true

9.65.3.2 bool QEComboBox::writeOnChange [read, write, protected]

Sets if this widget writes any changes as the user selects values (the QComboBox 'activated' signal is emitted). Default is 'true' (writes any changes when the QComboBox 'activated' signal is emitted).

9.65.4 Property Documentation

9.65.4.1 bool QEComboBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.65.4.2 bool QEComboBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.65.4.3 unsigned QEComboBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.65.4.4 QString QEComboBox::localEnumeration [read, write]

Enumrations values used when useDbEnumerations is false.

9.65.4.5 bool QEComboBox::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.65.4.6 UserLevels QEComboBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user

mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.65.4.7 QString QEComboBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.65.4.8 QString QEComboBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.65.4.9 QString QEComboBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.65.4.10 UserLevels QEComboBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.65.4.11 QString QEComboBox::variable [read, write]

EPICS variable name (CA PV)

9.65.4.12 bool QEComboBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.65.4.13 QString QEComboBox::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.65.4.14 bool QEComboBox::visible [read, write]

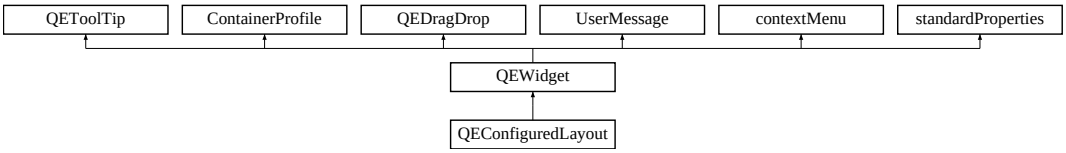
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEComboBox/QEComboBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEComboBox/QEComboBox.cpp

9.66 QEConfiguredLayout Class Reference

Inheritance diagram for QEConfiguredLayout:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }

Public Member Functions

- **QEConfiguredLayout** (QWidget *pParent=0, bool pSubscription=true)
- void **setItemDescription** (QString pValue)
- QString **getItemDescription** ()
- void **setShowItemList** (bool pValue)
- bool **getShowItemList** ()
- void **setConfigurationType** (int pValue)
- int **getConfigurationType** ()
- void **setConfigurationFile** (QString pValue)
- QString **getConfigurationFile** ()
- void **setConfigurationText** (QString pValue)
- QString **getConfigurationText** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- void **refreshFields** ()
- void **userLevelChanged** (userLevelTypes::userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)

- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

Public Attributes

- QList< _Item * > **itemList**
- QList< _Field * > **currentFieldList**

Protected Attributes

- QLabel * **qLabelItemDescription**
- QComboBox * **qComboBoxItemList**
- QVBoxLayout * **qVBoxLayoutFields**
- QScrollArea * **qScrollArea**
- QString **configurationFile**
- QString **configurationText**
- int **configurationType**
- int **detailsLayout**
- int **currentUserType**
- bool **subscription**

Properties

- QString **itemDescription**
- bool **showItemList**
- configurationTypesProperty **configurationType**
- detailsLayoutProperty **detailsLayout**
- userTypesProperty **currentUserType**

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.67 QEConfiguredLayoutManager Class Reference

Public Member Functions

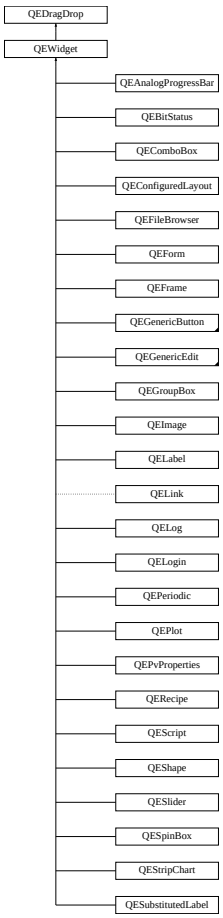
- **QEConfiguredLayoutManager** (QObject ∗pParent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **createWidget** (QWidget ∗pParent)
- void **initialize** (QDesignerFormEditorInterface ∗pCore)

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayoutManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayoutManager.cpp

9.68 QEDragDrop Class Reference

Inheritance diagram for QEDragDrop:



Public Member Functions

- **QEDragDrop** (QWidget ∗ownerIn)

Protected Member Functions

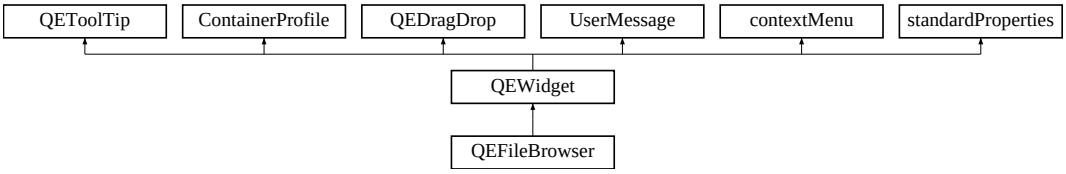
- void **qcaDragEnterEvent** (QDragEnterEvent ∗event)
- void **qcaDropEvent** (QDropEvent ∗event)
- void **qcaMousePressEvent** (QMouseEvent ∗event)
- virtual void **setDrop** (QVariant)
- virtual QVariant **getDrop** ()
- void **setAllowDrop** (bool allowDropIn)
- bool **getAllowDrop** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEDragDrop.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEDragDrop.cpp

9.69 QEFileBrowser Class Reference

Inheritance diagram for QEFileBrowser:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEFileBrowser** (QWidget ∗pParent=0)
- void **setDirectoryPath** (QString pValue)
- QString **getDirectoryPath** ()
- void **setShowDirectoryPath** (bool pValue)
- bool **getShowDirectoryPath** ()
- void **setShowDirectoryBrowser** (bool pValue)
- bool **getShowDirectoryBrowser** ()
- void **setShowRefresh** (bool pValue)
- bool **getShowRefresh** ()
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnSize** (bool pValue)
- bool **getShowColumnSize** ()
- void **setShowColumnFilename** (bool pValue)
- bool **getShowColumnFilename** ()
- void **setShowFileExtension** (bool pValue)
- bool **getShowFileExtension** ()
- void **setFileFilter** (QString pValue)
- QString **getFileFilter** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **updateTable** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()

Protected Attributes

- QLineEdit * **qlineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetFileBrowser * **qTableWidgetFileBrowser**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

Properties

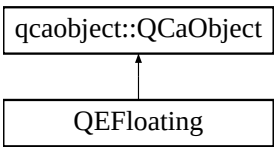
- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**
- detailsLayoutProperty **detailsLayout**

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

- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.70 QEFloating Class Reference

Inheritance diagram for QEFloating:



Public Slots

- void **writeFloating** (const double &data)

Signals

- void **floatingConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **floatingChanged** (const double &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **floatingArrayChanged** (const QVector< double > &values, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEFloating** (QString recordName, QObject *eventObject, QEFloatingFormatting *floatingFormattingIn, unsigned int variableIndexIn)
- **QEFloating** (QString recordName, QObject *eventObject, QEFloatingFormatting *floatingFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)

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

- /tmp/epicsqt/trunk/framework/data/include/QEFloating.h
- /tmp/epicsqt/trunk/framework/data/src/QEFloating.cpp

9.71 QEFloatingFormatting Class Reference

Public Types

- enum **formats** {
  FORMAT_e = 'e', **FORMAT_E** = 'E', **FORMAT_f** = 'f', **FORMAT_g** = 'g',
  FORMAT_G = 'G' }

Public Member Functions

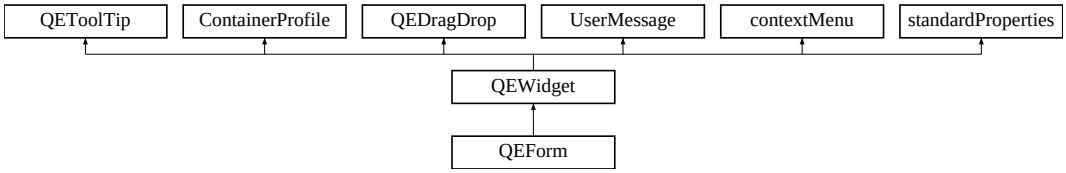
- double **formatFloating** (const QVariant &value)
- QVector< double > **formatFloatingArray** (const QVariant &value)
- QVariant **formatValue** (const double &floatingValue, generic::generic_types valueType)
- void **setPrecision** (unsigned int precision)
- void **setFormat** (formats format)
- unsigned int **getPrecision** ()
- int **getFormat** ()

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

- /tmp/epicsqt/trunk/framework/data/include/QEFloatingFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEFloatingFormatting.cpp

9.72 QEForm Class Reference

Inheritance diagram for QEForm:



Public Types

- enum **creationOptions** { **CREATION_OPTION_OPEN**, **CREATION_OPTION_-NEW_TAB**, **CREATION_OPTION_NEW_WINDOW** }
- enum **MessageFilterOptions** { **Match** = UserMessage::MESSAGE_FILTER_-MATCH, **None** = UserMessage::MESSAGE_FILTER_NONE }

Public Slots

- bool **readUiFile** ()
- void **launchGui** (QString guiName, QEForm::creationOptions createOption)

Public Member Functions

- **QEForm** (QWidget ∗parent=0)
- **QEForm** (const QString &uifileNameIn, QWidget ∗parent=0)
- void **commonInit** (const bool alertIfUINoFoundIn)
- QString **getQEGuiTitle** ()
- QString **getFullFileName** ()
- void setVariableNameAndSubstitutions (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- void **setUiFileName** (QString uiFile)
- QString **getUiFileName** ()
- void **setHandleGuiLaunchRequests** (bool handleGuiLaunchRequests)
- bool **getHandleGuiLaunchRequests** ()
- void **setResizeContents** (bool resizeContentsIn)
- bool **getResizeContents** ()
- QString **getContainedFrameworkVersion** ()
- QString **getUniqueIdentifier** ()
- void **setUniqueIdentifier** (QString name)
- void **setVariableNameSubstitutionsProperty** (QString variableNameSubstitutions)
- QString **getVariableNameSubstitutionsProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()

- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Member Functions

- void **setVariableNameSubstitutions** (QString variableNameSubstitutionsIn)

Protected Attributes

- QString **uiFileName**
- QString **fullUiFileName**
- bool **handleGuiLaunchRequests**
- bool **resizeContents**

Properties

- QString **uiFile**
- QString **variableSubstitutions**
- unsigned **int**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**

9.72.1 Member Function Documentation

9.72.1.1 void QEForm::setVariableNameAndSubstitutions (QString *variableNameIn*, QString *variableNameSubstitutionsIn*, unsigned int *variableIndex*) [virtual]

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

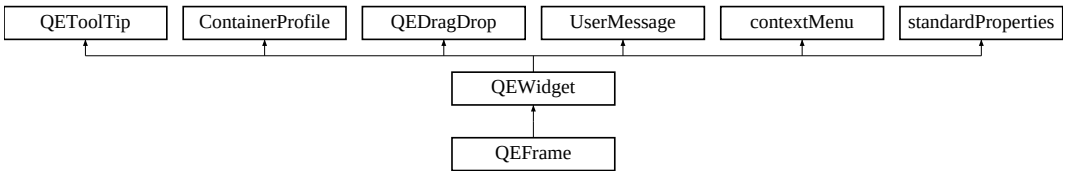
Reimplemented from QEWidget.

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

- /tmp/epicsqt/trunk/framework/widgets/QEForm/QEForm.h
- /tmp/epicsqt/trunk/framework/widgets/QEForm/QEForm.cpp

9.73 QEFrame Class Reference

Inheritance diagram for QEFrame:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- **QEFrame** (QWidget *parent=0)
- QSize **sizeHint** () const

Properties

- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.73.1 Member Enumeration Documentation

9.73.1.1 enum QEFrame::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.73.2 Property Documentation

9.73.2.1 bool QEFrame::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.73.2.2 bool QEFrame::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.73.2.3 unsigned QEFrame::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.73.2.4 UserLevels QEFrame::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.73.2.5 QString QEFrame::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.2.6 QString QEFrame::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.2.7 QString QEFrame::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.2.8 UserLevels QEFrame::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.73.2.9 bool QEFrame::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.73.2.10 bool QEFrame::visible [read, write]

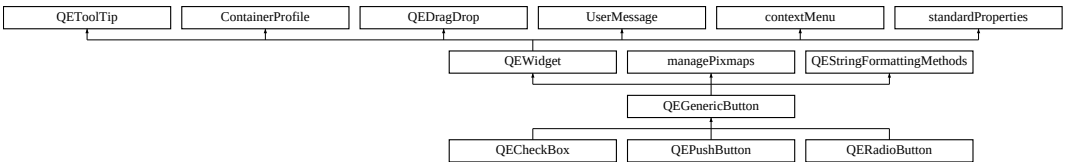
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEFrame/QEFrame.h
- /tmp/epicsqt/trunk/framework/widgets/QEFrame/QEFrame.cpp

9.74 QEGenericButton Class Reference

Inheritance diagram for QEGenericButton:



Public Types

- enum **updateOptions** { **UPDATE_TEXT**, **UPDATE_ICON**, **UPDATE_TEXT_AND_-ICON**, **UPDATE_STATE** }

Public Member Functions

- **QEGenericButton** (QWidget ∗owner)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUpdateOption** (updateOptions updateOptionIn)
- updateOptions **getUpdateOption** ()
- void **setTextAlignment** (Qt::Alignment alignment)
- Qt::Alignment **getTextAlignment** ()
- void **setPassword** (QString password)
- QString **getPassword** ()
- void **setConfirmAction** (bool confirmRequiredIn)
- bool **getConfirmAction** ()
- void **setWriteOnPress** (bool writeOnPress)
- bool **getWriteOnPress** ()
- void **setWriteOnRelease** (bool writeOnRelease)
- bool **getWriteOnRelease** ()
- void **setWriteOnClick** (bool writeOnClick)

- bool **getWriteOnClick** ()
- void **setPressText** (QString pressText)
- QString **getPressText** ()
- void **setReleaseText** (QString releaseTextIn)
- QString **getReleaseText** ()
- void **setClickText** (QString clickTextIn)
- QString **getClickText** ()
- void **setClickCheckedText** (QString clickCheckedTextIn)
- QString **getClickCheckedText** ()
- void **setProgram** (QString program)
- QString **getProgram** ()
- void **setArguments** (QStringList arguments)
- QStringList **getArguments** ()
- void **setGuiName** (QString guiName)
- QString **getGuiName** ()
- void **setPrioritySubstitutions** (QString prioritySubstitutionsIn)
- QString **getPrioritySubstitutions** ()
- void **setCreationOption** (QEForm::creationOptions creationOption)
- QEForm::creationOptions **getCreationOption** ()
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()

Protected Member Functions

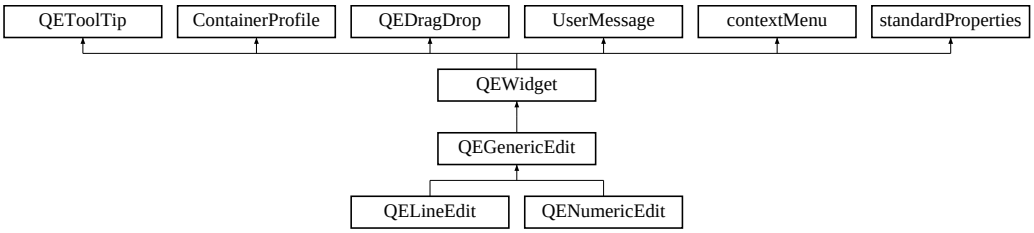
- void **connectionChanged** (QCaConnectionInfo &connectionInfo)
- void **setGenericButtonText** (const QString &text, QCaAlarmInfo &alarmInfo, QCaDateTime &, const unsigned int &variableIndex)
- void **userPressed** ()
- void **userReleased** ()
- void **userClicked** (bool checked)
- void **launchGui** (QString guiName, QEForm::creationOptions creationOption)
- virtual updateOptions **getDefaultUpdateOption** ()=0
- void **setup** ()
- void **establishConnection** (unsigned int variableIndex)

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEGenericButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEGenericButton.cpp

9.75 QEGenericEdit Class Reference

Inheritance diagram for QEGenericEdit:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void userChange (const QVariant &oldValue, const QVariant &newValue, const QVariant &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- QEGenericEdit (QWidget ∗parent=0)
- QEGenericEdit (const QString &variableName, QWidget ∗parent=0)
- void setWriteOnLoseFocus (bool writeOnLoseFocus)

- bool getWriteOnLoseFocus ()
- void setWriteOnEnter (bool writeOnEnter)
- bool getWriteOnEnter ()
- void setWriteOnFinish (bool writeOnFinish)
- bool getWriteOnFinish ()
- void setConfirmWrite (bool confirmWrite)
- bool getConfirmWrite ()
- void setSubscribe (bool subscribe)
- bool getSubscribe ()
- void **writeValue** (qcaobject::QCaObject *qca, QVariant newValue)

Protected Member Functions

- void **setDataIfNoFocus** (const QVariant &value, QCaAlarmInfo &alarmInfo, QCaDateTime &dateTime)
- bool **getIsConnected** ()
- bool **testAndClearIsFirstUpdate** ()
- virtual void **setValue** (const QVariant &value)=0
- virtual QVariant **getValue** ()=0
- virtual bool **writeData** (const QVariant &value, QString &message)=0
- void writeNow ()

  Write the value now.

Protected Attributes

- QVariant **lastValue**
- QVariant **lastUserValue**
- bool **messageDialogPresent**
- bool **writeFailMessageDialogPresent**
- bool **isConnected**

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool writeOnLoseFocus
- bool writeOnEnter
- bool writeOnFinish
- bool confirmWrite
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle

- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.75.1 Member Enumeration Documentation

9.75.1.1 enum QEGenericEdit::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.75.2 Constructor & Destructor Documentation

9.75.2.1 QEGenericEdit::QEGenericEdit (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.75.2.2 QEGenericEdit::QEGenericEdit (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.75.3 Member Function Documentation

9.75.3.1 bool QEGenericEdit::getConfirmWrite ( )

Returns 'true' if this widget will ask for confirmation (using a dialog box) prior to writing data.

9.75.3.2 bool QEGenericEdit::getSubscribe ( )

Returns 'true' if this widget subscribes for data updates and displays current data.

9.75.3.3 bool QEGenericEdit::getWriteOnEnter ( )

Returns 'true' if this widget writes any changes when the user presses 'enter'.

9.75.3.4 bool QEGenericEdit::getWriteOnFinish ( )

Returns 'true' if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted).

9.75.3.5 bool QEGenericEdit::getWriteOnLoseFocus ( )

Returns 'true' if this widget automatically writes any changes when it loses focus.

9.75.3.6 void QEGenericEdit::setConfirmWrite (bool *confirmWrite*)

Sets if this widget will ask for confirmation (using a dialog box) prior to writing data. Default is 'false' (will not ask for confirmation (using a dialog box) prior to writing data).

9.75.3.7 void QEGenericEdit::setSubscribe (bool *subscribe*)

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

9.75.3.8 void QEGenericEdit::setWriteOnEnter (bool *writeOnEnter*)

Sets if this widget writes any changes when the user presses 'enter'. Note, the current value will be written even if the user has not changed it. Default is 'true' (writes any changes when the user presses 'enter').

9.75.3.9 void QEGenericEdit::setWriteOnFinish (bool *writeOnFinish*)

Sets if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted). No writing occurs if no changes were made. Default is 'true' (writes any changes when the QLineEdit 'editingFinished' signal is emitted).

9.75.3.10 void QEGenericEdit::setWriteOnLoseFocus (bool *writeOnLoseFocus*)

Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

9.75.4 Property Documentation

9.75.4.1 bool QEGenericEdit::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.75.4.2 bool QEGenericEdit::confirmWrite [read, write]

Sets if this widget will ask for confirmation (using a dialog box) prior to writing data. Default is 'false' (will not ask for confirmation (using a dialog box) prior to writing data).

9.75.4.3 bool QEGenericEdit::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.75.4.4 unsigned QEGenericEdit::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Reimplemented in QELineEdit.

9.75.4.5 bool QEGenericEdit::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.75.4.6 UserLevels QEGenericEdit::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.75.4.7 QString QEGenericEdit::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.75.4.8 QString QEGenericEdit::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.75.4.9 QString QEGenericEdit::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.75.4.10 UserLevels QEGenericEdit::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.75.4.11 QString QEGenericEdit::variable [read, write]

EPICS variable name (CA PV)

9.75.4.12 bool QEGenericEdit::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.75.4.13 QString QEGenericEdit::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.75.4.14 bool QEGenericEdit::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.75.4.15 bool QEGenericEdit::writeOnEnter [read, write]

Sets if this widget writes any changes when the user presses 'enter'. Note, the current value will be written even if the user has not changed it. Default is 'true' (writes any changes when the user presses 'enter').

9.75.4.16 bool QEGenericEdit::writeOnFinish [read, write]

Sets if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted). No writing occurs if no changes were made. Default is 'true' (writes any changes when the QLineEdit 'editingFinished' signal is emitted).

9.75.4.17 bool QEGenericEdit::writeOnLoseFocus [read, write]

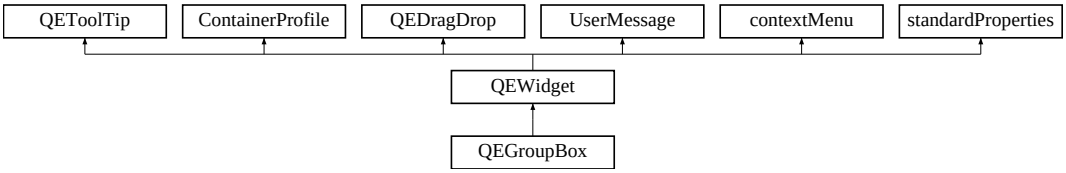
Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QEGenericEdit.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QEGenericEdit.cpp

9.76 QEGroupBox Class Reference

Inheritance diagram for QEGroupBox:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- **QEGroupBox** (QWidget *parent=0)
- QSize **sizeHint** () const

Properties

- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.76.1 Member Enumeration Documentation

9.76.1.1 enum QEGroupBox::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.76.2 Property Documentation

9.76.2.1 bool QEGroupBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.76.2.2 bool QEGroupBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.76.2.3 unsigned QEGroupBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.76.2.4 UserLevels QEGroupBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.76.2.5 QString QEGroupBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.76.2.6 QString QEGroupBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.76.2.7 QString QEGroupBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.76.2.8 UserLevels QEGroupBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.76.2.9 bool QEGroupBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.76.2.10 bool QEGroupBox::visible [read, write]

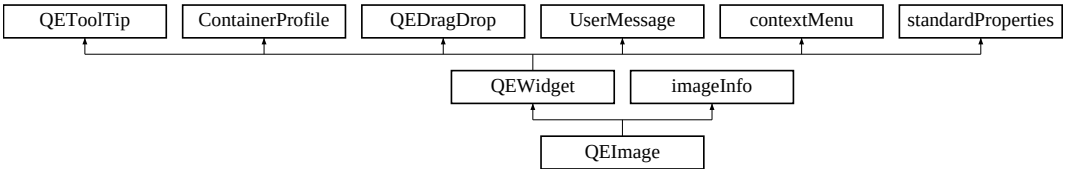
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEGroupBox/QEGroupBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEGroupBox/QEGroupBox.cpp

9.77 QEImage Class Reference

Inheritance diagram for QEImage:



Classes

- struct **rgbPixel**

Public Types

- enum selectOptions {
  SO_NONE, SO_PANNING, SO_VSLICE, SO_HSLICE,
  SO_AREA1, **SO_AREA2**, **SO_AREA3**, SO_AREA4,
  SO_PROFILE, SO_TARGET, SO_BEAM }
- enum formatOptions { GREY8, GREY12, GREY16, RGB_888 }
- enum resizeOptions { RESIZE_OPTION_ZOOM, RESIZE_OPTION_FIT }
- enum rotationOptions { ROTATION_0, ROTATION_90_RIGHT, ROTATION_90_-LEFT, ROTATION_180 }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum FormatOptions { Grey_8 = QEImage::GREY8, Grey_12 = QEImage::GREY12, Grey_16 = QEImage::GREY16, **RGB** = QEImage::RGB_888 }
- enum ResizeOptions { Zoom = QEImage::RESIZE_OPTION_ZOOM, Fit = QEImage::RESIZE_-OPTION_FIT }
- enum RotationOptions { NoRotation = QEImage::ROTATION_0, Rotate90Right = QEImage::ROTATION_90_RIGHT, Rotate90Left = QEImage::ROTATION_90_-LEFT, Rotate180 = QEImage::ROTATION_180 }

Public Slots

- void setSelectPanMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectVSliceMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectHSliceMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectArea1Mode ()

Framework use only. Slot to allow external setting of selection menu options.

- void setSelectArea2Mode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectArea3Mode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectArea4Mode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectProfileMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectTargetMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void setSelectBeamMode ()

  Framework use only. Slot to allow external setting of selection menu options.
- void pauseClicked ()

  Framework use only. Slot to allow external setting of selection menu options.
- void saveClicked ()

  Framework use only. Slot to allow external setting of selection menu options.
- void roi1Changed ()

  Framework use only. Slot to allow external setting of selection menu options.
- void roi2Changed ()

  Framework use only. Slot to allow external setting of selection menu options.
- void roi3Changed ()

  Framework use only. Slot to allow external setting of selection menu options.
- void roi4Changed ()

  Framework use only. Slot to allow external setting of selection menu options.
- void targetClicked ()

  Framework use only. Slot to allow external setting of selection menu options.

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- QEImage (QWidget *parent=0)
- QEImage (const QString &variableName, QWidget *parent=0)
- ~QEImage ()

  Destructor.
- selectOptions **getSelectionOption** ()
- void setFormatOption (formatOptions formatOption)

Access function for #formatOption property - refer to #formatOption property for details.

- formatOptions getFormatOption ()

  Access function for #formatOption property - refer to #formatOption property for details.

- void setResizeOption (resizeOptions resizeOptionIn)

  Access function for #resizeOption property - refer to #resizeOption property for details.

- resizeOptions getResizeOption ()

  Access function for #resizeOption property - refer to #resizeOption property for details.

- void setZoom (int zoomIn)

  Access function for zoom property - refer to zoom property for details.

- int getZoom ()

  Access function for zoom property - refer to zoom property for details.

- void setRotation (rotationOptions rotationIn)

  Access function for #rotation property - refer to #rotation property for details.

- rotationOptions getRotation ()

  Access function for #rotation property - refer to #rotation property for details.

- void setHorizontalFlip (bool flipHozIn)

  Access function for horizontalFlip property - refer to horizontalFlip property for details.

- bool getHorizontalFlip ()

  Access function for horizontalFlip property - refer to horizontalFlip property for details.

- void setVerticalFlip (bool flipVertIn)

  Access function for verticalFlip property - refer to verticalFlip property for details.

- bool getVerticalFlip ()

  Access function for verticalFlip property - refer to verticalFlip property for details.

- void setInitialHozScrollPos (int initialHosScrollPosIn)

  Access function for initialHosScrollPos property - refer to initialHosScrollPos property for details.

- int getInitialHozScrollPos ()

  Access function for initialHosScrollPos property - refer to initialHosScrollPos property for details.

- void setInitialVertScrollPos (int initialVertScrollPosIn)

  Access function for initialVertScrollPos property - refer to initialVertScrollPos property for details.

- int getInitialVertScrollPos ()

  Access function for initialVertScrollPos property - refer to initialVertScrollPos property for details.

- void setDisplayButtonBar (bool displayButtonBarIn)

  Access function for displayButtonBar property - refer to displayButtonBar property for details.

- bool getDisplayButtonBar ()

  Access function for displayButtonBar property - refer to displayButtonBar property for details.

- void setShowTime (bool pValue)

  Access function for showTime property - refer to showTime property for details.

- bool getShowTime ()

  Access function for showTime property - refer to showTime property for details.
- void setVertSliceMarkupColor (QColor pValue)

  Access function for vertSliceColor property - refer to vertSliceColor property for details.
- QColor getVertSliceMarkupColor ()

  Access function for vertSliceColor property - refer to vertSliceColor property for details.
- void setHozSliceMarkupColor (QColor pValue)

  Access function for hozSliceColor property - refer to hozSliceColor property for details.
- QColor getHozSliceMarkupColor ()

  Access function for hozSliceColor property - refer to hozSliceColor property for details.
- void setProfileMarkupColor (QColor pValue)

  Access function for profileColor property - refer to profileColor property for details.
- QColor getProfileMarkupColor ()

  Access function for profileColor property - refer to profileColor property for details.
- void setAreaMarkupColor (QColor pValue)

  Access function for areaColor property - refer to areaColor property for details.
- QColor getAreaMarkupColor ()

  Access function for areaColor property - refer to areaColor property for details.
- void setTargetMarkupColor (QColor pValue)

  Access function for targetColor property - refer to targetColor property for details.
- QColor getTargetMarkupColor ()

  Access function for targetColor property - refer to targetColor property for details.
- void setBeamMarkupColor (QColor pValue)

  Access function for beamColor property - refer to beamColor property for details.
- QColor getBeamMarkupColor ()

  Access function for beamColor property - refer to beamColor property for details.
- void setTimeMarkupColor (QColor pValue)

  Access function for timeColor property - refer to timeColor property for details.
- QColor getTimeMarkupColor ()

  Access function for timeColor property - refer to timeColor property for details.
- void setDisplayCursorPixelInfo (bool displayCursorPixelInfoIn)

  Access function for #displayCursorPixelInfo property - refer to #displayCursorPixelInfo property for details.
- bool getDisplayCursorPixelInfo ()

  Access function for #displayCursorPixelInfo property - refer to #displayCursorPixelInfo property for details.
- void setContrastReversal (bool contrastReversalIn)

  Access function for #contrastReversal property - refer to #contrastReversal property for details.
- bool getContrastReversal ()

  Access function for #contrastReversal property - refer to #contrastReversal property for details.
- void setEnableVertSliceSelection (bool enableVSliceSelectionIn)

Access function for enableVertSliceSelection property - refer to enableVertSliceSelection property for details.

- bool getEnableVertSliceSelection ()

  Access function for enableVertSliceSelection property - refer to enableVertSliceSelection property for details.

- void setEnableHozSliceSelection (bool enableHSliceSelectionIn)

  Access function for enableHozSliceSelection property - refer to enableHozSliceSelection property for details.

- bool getEnableHozSliceSelection ()

  Access function for enableHozSliceSelection property - refer to enableHozSliceSelection property for details.

- void setEnableAreaSelection (bool enableAreaSelectionIn)

  Access function for #enableAreaSelection property - refer to #enableAreaSelection property for details.

- bool getEnableAreaSelection ()

  Access function for #enableAreaSelection property - refer to #enableAreaSelection property for details.

- void setEnableProfileSelection (bool enableProfileSelectionIn)

  Access function for #enableProfileSelection property - refer to #enableProfileSelection property for details.

- bool getEnableProfileSelection ()

  Access function for #enableProfileSelection property - refer to #enableProfileSelection property for details.

- void setEnableTargetSelection (bool enableTargetSelectionIn)

  Access function for #enableTargetSelection property - refer to #enableTargetSelection property for details.

- bool getEnableTargetSelection ()

  Access function for #enableTargetSelection property - refer to #enableTargetSelection property for details.

- void setEnableBrightnessContrast (bool enableBrightnessContrastIn)

  Access function for enableBrightnessContrast property - refer to enableBrightnessContrast property for details.

- bool getEnableBrightnessContrast ()

  Access function for enableBrightnessContrast property - refer to enableBrightnessContrast property for details.

- void setAutoBrightnessContrast (bool autoBrightnessContrastIn)

  Access function for autoBrightnessContrast property - refer to autoBrightnessContrast property for details.

- bool getAutoBrightnessContrast ()

  Access function for autoBrightnessContrast property - refer to autoBrightnessContrast property for details.

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.

- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.

- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatOptionProperty (FormatOptions formatOption)

  Access function for #formatOption property - refer to #formatOption property for details.
- FormatOptions getFormatOptionProperty ()

  Access function for #formatOption property - refer to #formatOption property for details.
- void setResizeOptionProperty (ResizeOptions resizeOption)

  Access function for #resizeOption property - refer to #resizeOption property for details.
- ResizeOptions getResizeOptionProperty ()

  Access function for #resizeOption property - refer to #resizeOption property for details.
- void setRotationProperty (RotationOptions rotation)

  Access function for #rotation property - refer to #rotation property for details.
- RotationOptions getRotationProperty ()

  Access function for #rotation property - refer to #rotation property for details.

Protected Types

- enum **variableIndexes** {

  IMAGE_VARIABLE, **WIDTH_VARIABLE**, **HEIGHT_VARIABLE**, **ROI1_X_VARIABLE**,

  ROI1_Y_VARIABLE, **ROI1_W_VARIABLE**, **ROI1_H_VARIABLE**, **ROI2_X_VARIABLE**,

  ROI2_Y_VARIABLE, **ROI2_W_VARIABLE**, **ROI2_H_VARIABLE**, **ROI3_X_VARIABLE**,

  ROI3_Y_VARIABLE, **ROI3_W_VARIABLE**, **ROI3_H_VARIABLE**, **ROI4_X_VARIABLE**,

  ROI4_Y_VARIABLE, **ROI4_W_VARIABLE**, **ROI4_H_VARIABLE**, **TARGET_X_-VARIABLE**,

  TARGET_Y_VARIABLE, **BEAM_X_VARIABLE**, **BEAM_Y_VARIABLE**, **TARGET_-TRIGGER_VARIABLE**,

  CLIPPING_ONOFF_VARIABLE, **CLIPPING_LOW_VARIABLE**, **CLIPPING_HIGH_-VARIABLE**, **QEIMAGE_NUM_VARIABLES** }

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()

- QVariant **copyData** ()
- void **paste** (QVariant v)
- void **resizeEvent** (QResizeEvent ∗)

Protected Attributes

- QEIntegerFormatting **integerFormatting**
- resizeOptions **resizeOption**
- int zoom

  Zoom percentage. Used when #resizeOption is Zoom.
- rotationOptions **rotation**
- bool **flipVert**
- bool **flipHoz**
- int **initialHozScrollPos**
- int initialVertScrollPos
- bool displayButtonBar
- bool enableBrightnessContrast
- bool autoBrightnessContrast

Properties

- QString imageVariable
- QString widthVariable
- QString heightVariable
- QString regionOfInterest1XVariable
- QString regionOfInterest1YVariable
- QString regionOfInterest1WVariable
- QString regionOfInterest1HVariable
- QString regionOfInterest2XVariable
- QString regionOfInterest2YVariable
- QString regionOfInterest2WVariable
- QString regionOfInterest2HVariable
- QString regionOfInterest3XVariable
- QString regionOfInterest3YVariable
- QString regionOfInterest3WVariable
- QString regionOfInterest3HVariable
- QString regionOfInterest4XVariable
- QString regionOfInterest4YVariable
- QString regionOfInterest4WVariable
- QString regionOfInterest4HVariable
- QString targetXVariable
- QString targetYVariable
- QString beamXVariable
- QString beamYVariable
- QString targetTriggerVariable

- QString clippingOnOffVariable
- QString clippingLowVariable
- QString clippingHighVariable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- FormatOptions formatOption
- bool enableVertSliceSelection
- bool enableHozSliceSelection
- bool showTime
- QColor vertSliceColor
- QColor hozSliceColor
- QColor profileColor
- QColor areaColor
- QColor beamColor
- QColor targetColor
- QColor timeColor
- ResizeOptions resizeOption
- RotationOptions rotation
- bool verticalFlip
- bool horizontalFlip
- int initialHosScrollPos

9.77.1 Member Enumeration Documentation

9.77.1.1 enum QEImage::formatOptions

Video format options

Enumerator:

GREY8 8 bit grey scale

GREY12 12 bit grey scale

GREY16 16 bit grey scale

RGB_888 24 bit RGB

9.77.1.2 enum QEImage::FormatOptions

User friendly enumerations for #formatOption property - refer to #formatOption property and formatOptions enumeration for details.

Enumerator:

Grey_8 8 bit grey scale

Grey_12 12 bit grey scale

Grey_16 16 bit grey scale

9.77.1.3 enum QEImage::ResizeOptions

User friendly enumerations for #resizeOption property

Enumerator:

Zoom Zoom to selected percentage.

Fit Zoom to fit the current window size.

9.77.1.4 enum QEImage::resizeOptions

Image resize options

Enumerator:

RESIZE_OPTION_ZOOM Zoom to selected percentage.

RESIZE_OPTION_FIT Zoom to fit the current window size.

9.77.1.5 enum QEImage::rotationOptions

Image rotation options

Enumerator:

ROTATION_0 No image rotation.

ROTATION_90_RIGHT Rotate image 90 degrees clockwise.

ROTATION_90_LEFT Rotate image 90 degrees anticlockwise.

ROTATION_180 Rotate image 180 degrees.

9.77.1.6 enum QEImage::RotationOptions

User friendly enumerations for #rotation property

Enumerator:

NoRotation No image rotation.

Rotate90Right Rotate image 90 degrees clockwise.

Rotate90Left Rotate image 90 degrees anticlockwise.

Rotate180 Rotate image 180 degrees.

9.77.1.7 enum QEImage::selectOptions

Internal use only. Selection options. What will happen when the user interacts with the image area

Enumerator:

SO_NONE Do nothing.

SO_PANNING User is panning.

SO_VSLICE Select the vertical slice point.

SO_HSLICE Select the horizontal slice point.

SO_AREA4 User is selecting an area (for region of interest)

SO_PROFILE Select an arbitrary line across the image (to determine a profile)

SO_TARGET Mark the target point.

SO_BEAM Mark the current beam location.

9.77.1.8 enum QEImage::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.77.2 Constructor & Destructor Documentation

9.77.2.1 QEImage::QEImage (QWidget * *parent* = 0)

Create without a variable. Use setVariableName'n'Property() - where 'n' is a number from 0 to 26 - and setSubstitutionsProperty() to define variables and, optionally, macro substitutions later. Note, each variable property is named by function (such as imageVariable and widthVariable) but given a numeric get and set property access function such as setVariableName22Property(). Refer to the property definitions to determine what 'set' and 'get' function is used for each varible, or use Qt library functions to set or get the variable names by name.

9.77.2.2 QEImage::QEImage (const QString & *variableName,* QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. The variable is set up as the first variable. This is consistant with other widgets, but will not result in an updating image as the width and height variables are required as a minimum.

9.77.3 Member Function Documentation

9.77.3.1 void QEImage::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.77.4 Member Data Documentation

9.77.4.1 bool QEImage::autoBrightnessContrast [read, write, protected]

If true, local brightness and contrast controls are displayed. The brightness and contrast is set to use the full range of pixels in the selected area.

9.77.4.2 bool QEImage::displayButtonBar [read, write, protected]

If true, a button bar will be displayed above the image. If not displayed, all buttons in the button bar are still available in the right click menu.

9.77.4.3 bool QEImage::enableBrightnessContrast [read, write, protected]

If true, auto set local brightness and contrast when any area is selected. The brightness and contrast is set to use the full range of pixels in the selected area.

9.77.4.4 int QEImage::initialVertScrollPos [read, write, protected]

Sets the initial position of the vertical scroll bar, if pressent. Used to set up an initial view when zoomed in.

9.77.5 Property Documentation

9.77.5.1 bool QEImage::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.77.5.2 QColor QEImage::areaColor [read, write]

Used to select the color of the area selection markups.

9.77.5.3 QColor QEImage::beamColor [read, write]

Used to select the color of the beam marker.

9.77.5.4 QString QEImage::beamXVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected beam X position.

9.77.5.5 QString QEImage::beamYVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected beam Y position.

9.77.5.6 QString QEImage::clippingHighVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping high level.

9.77.5.7 QString QEImage::clippingLowVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping low level.

9.77.5.8 QString QEImage::clippingOnOffVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping on/off command.

9.77.5.9 bool QEImage::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.77.5.10 bool QEImage::enableHozSliceSelection [read, write]

If true, the option to select a horizontal slice through the image will be available to the user. This will be used to generate a horizontal pixel profile.

9.77.5.11 bool QEImage::enableVertSliceSelection [read, write]

If true, the option to select a vertical slice through the image will be available to the user. This will be used to generate a vertical pixel profile.

9.77.5.12 FormatOptions QEImage::formatOption [read, write]

Video format. EPICS data type size will typically be adequate for the number of bits required (one byte for 8 bits, 2 bytes for 12 and 16 bits), but can be larger (4 bytes for 24 bits.)

9.77.5.13 QString QEImage::heightVariable [read, write]

EPICS variable name (CA PV). This variable is used to read the height of the image.

9.77.5.14 bool QEImage::horizontalFlip [read, write]

If true, flip image horizontally.

9.77.5.15 QColor QEImage::hozSliceColor [read, write]

Used to select the color of the horizontal slice markup.

9.77.5.16 QString QEImage::imageVariable [read, write]

EPICS variable name (CA PV). This variable is used as the source the image waveform.

9.77.5.17 int QEImage::initialHosScrollPos [read, write]

Sets the initial position of the horizontal scroll bar, if pressent. Used to set up an initial view when zoomed in.

9.77.5.18 unsigned QEImage::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.77.5.19 QColor QEImage::profileColor [read, write]

Used to select the color of the arbitrarty profile line markup.

9.77.5.20 QString QEImage::regionOfInterest1HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest height.

9.77.5.21 QString QEImage::regionOfInterest1WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest width.

9.77.5.22 QString QEImage::regionOfInterest1XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest X position.

9.77.5.23 QString QEImage::regionOfInterest1YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest Y position.

9.77.5.24 QString QEImage::regionOfInterest2HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest height.

9.77.5.25 QString QEImage::regionOfInterest2WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest width.

9.77.5.26 QString QEImage::regionOfInterest2XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest X position.

9.77.5.27 QString QEImage::regionOfInterest2YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest Y position.

9.77.5.28 QString QEImage::regionOfInterest3HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest height.

9.77.5.29 QString QEImage::regionOfInterest3WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest width.

9.77.5.30 QString QEImage::regionOfInterest3XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest X position.

9.77.5.31 QString QEImage::regionOfInterest3YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest Y position.

9.77.5.32 QString QEImage::regionOfInterest4HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest height.

9.77.5.33 QString QEImage::regionOfInterest4WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest width.

9.77.5.34 QString QEImage::regionOfInterest4XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest X position.

9.77.5.35 QString QEImage::regionOfInterest4YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest Y position.

9.77.5.36 ResizeOptions QEImage::resizeOption [read, write]

Resize option. Zoom to zoom to the percentage given by the zoom property, or fit to the window size.

9.77.5.37 RotationOptions QEImage::rotation [read, write]

Image rotation option.

9.77.5.38 bool QEImage::showTime [read, write]

If true, the image timestamp will be written in the top left of the image.

9.77.5.39 QColor QEImage::targetColor [read, write]

Used to select the color of the target marker.

9.77.5.40 QString QEImage::targetTriggerVariable [read, write]

EPICS variable name (CA PV). This variable is used to write a 'trigger' to initiate movement of the target into the beam as defined by the target and beam X and Y positions.

9.77.5.41 QString QEImage::targetXVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected target X position.

9.77.5.42 QString QEImage::targetYVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected target Y position.

9.77.5.43 QColor QEImage::timeColor [read, write]

Used to select the color of the timestamp.

9.77.5.44 UserLevels QEImage::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.77.5.45 QString QEImage::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.77.5.46 QString QEImage::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.77.5.47 QString QEImage::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.77.5.48 UserLevels QEImage::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.77.5.49 bool QEImage::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.77.5.50 QString QEImage::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'CAM=1, NAME = "Image 1"' These substitutions are applied to all the variable names.

9.77.5.51 bool QEImage::verticalFlip [read, write]

If true, flip image vertically.

9.77.5.52 QColor QEImage::vertSliceColor [read, write]

Used to select the color of the vertical slice markup.

9.77.5.53 bool QEImage::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.77.5.54 QString QEImage::widthVariable [read, write]

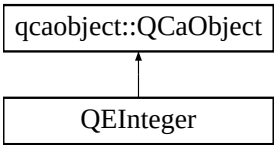
EPICS variable name (CA PV). This variable is used to read the width of the image.

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.cpp

9.78 QEInteger Class Reference

Inheritance diagram for QEInteger:



Public Slots

- void **writeInteger** (const long &data)

Signals

- void **integerConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **integerChanged** (const long &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **integerArrayChanged** (const QVector< long > &values, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEInteger** (QString recordName, QObject *eventObject, QEIntegerFormatting *integerFormattingIn, unsigned int variableIndexIn)
- **QEInteger** (QString recordName, QObject *eventObject, QEIntegerFormatting *integerFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)

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

- /tmp/epicsqt/trunk/framework/data/include/QEInteger.h
- /tmp/epicsqt/trunk/framework/data/src/QEInteger.cpp

9.79 QEIntegerFormatting Class Reference

`#include <QEIntegerFormatting.h>`

Public Member Functions

- QEIntegerFormatting ()

  Constructor.
- long formatInteger (const QVariant &value)
- QVector< long > formatIntegerArray (const QVariant &value)
- QVariant formatValue (const long &integerValue, generic::generic_types valueType)
- void setRadix (unsigned int radix)

  Set the radix used for all conversions. Default is 10.
- unsigned int getPrecision ()

  Get the precision used for all conversions.
- unsigned int getRadix ()

  Get the radix used for all conversions.

9.79.1 Detailed Description

This class holds formatting instructions and uses them to convert between an integer and a QVariant of any type. It is generally set up with it's formatting instructions and then passed to a QEInteger class that will sink and source integer data to widgets or other code. It is used to convert data to and from a QCaObject (which sources and sinks data in the form of a QVariant where the QVariant reflects the underlying variable data type) and the QEInteger class. An example of a requirement for integer data is a combo box which must determine an integer index to select a menu option.

9.79.2 Member Function Documentation

9.79.2.1 long QEIntegerFormatting::formatInteger (const QVariant & *value*)

Given a data value of any type, format it as an integer according to the formatting instructions held by the class. This is used to convert the QVariant value received from a QCaObject, which is still based on the data variable type, to an integer.

9.79.2.2 QVector< long > QEIntegerFormatting::formatIntegerArray (const QVariant & *value*)

Given a data value of any type, format it as an array of integers according to the formatting instructions held by the class. This is used to convert the QVariant value received from a QCaObject, which is still based on the data variable type, to an integer array. Typically used where the input QVariant value is an array of data values, but will work for any QVariant type.

9.79.2.3 QVariant QEIntegerFormatting::formatValue (const long & *integerValue,* generic::generic_types *valueType*)

Given an integer value, format it as a data value of the specified type, according to the formatting instructions held by the class. This is used when writing integer data to a QCaObject.

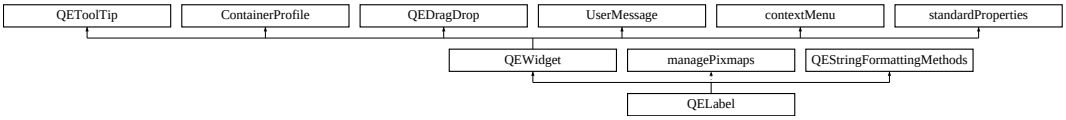
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/data/include/QEIntegerFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEIntegerFormatting.cpp

9.80 QELabel Class Reference

`#include <QELabel.h>`

Inheritance diagram for QELabel:



Public Types

- enum updateOptions { UPDATE_TEXT, UPDATE_PIXMAP }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_-FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInte-ger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_-LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_-AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QELabel::UPDATE_TEXT, Picture = QELabel::UPDATE_-PIXMAP }

  User friendly enumerations for updateOption property - refer to QELabel::updateOptions for details.

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- QELabel (QWidget *parent=0)
- QELabel (const QString &variableName, QWidget *parent=0)
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.
- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.
- void setUpdateOptionProperty (UpdateOptions updateOption)

  Access function for #updateOption property - refer to #updateOption property for details.
- UpdateOptions getUpdateOptionProperty ()

  Access function for #updateOption property - refer to #updateOption property for details.
- void setPixmap0Property (QPixmap pixmap)

  'Set' access function for pixmap0 properties. Refer to pixmap0 property for details
- void setPixmap1Property (QPixmap pixmap)

  'Set' access function for pixmap1 properties. Refer to pixmap1 property for details
- void setPixmap2Property (QPixmap pixmap)

  'Set' access function for pixmap2 properties. Refer to pixmap2 property for details
- void setPixmap3Property (QPixmap pixmap)

  'Set' access function for pixmap3 properties. Refer to pixmap3 property for details
- void setPixmap4Property (QPixmap pixmap)

  'Set' access function for pixmap4 properties. Refer to pixmap4 property for details
- void setPixmap5Property (QPixmap pixmap)

  'Set' access function for pixmap5 properties. Refer to pixmap5 property for details
- void setPixmap6Property (QPixmap pixmap)

  'Set' access function for pixmap6 properties. Refer to pixmap6 property for details
- void setPixmap7Property (QPixmap pixmap)

  'Set' access function for pixmap7 properties. Refer to pixmap7 property for details
- QPixmap getPixmap0Property ()

  'Get' access function for pixmap0 properties. Refer to pixmap0 property for details
- QPixmap getPixmap1Property ()

  'Get' access function for pixmap1 properties. Refer to pixmap1 property for details
- QPixmap getPixmap2Property ()

'Get' access function for pixmap2 properties. Refer to pixmap2 property for details

- QPixmap getPixmap3Property ()

  'Get' access function for pixmap3 properties. Refer to pixmap3 property for details

- QPixmap getPixmap4Property ()

  'Get' access function for pixmap4 properties. Refer to pixmap4 property for details

- QPixmap getPixmap5Property ()

  'Get' access function for pixmap5 properties. Refer to pixmap5 property for details

- QPixmap getPixmap6Property ()

  'Get' access function for pixmap6 properties. Refer to pixmap6 property for details

- QPixmap getPixmap7Property ()

  'Get' access function for pixmap7 properties. Refer to pixmap7 property for details

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7

9.80.1 Detailed Description

This class is a EPICS aware label widget based on the Qt label widget. When a variable is defined, the label text (or optionally the background pixmap) will be updated. The label will be disabled if the variable is invalid. It is tighly integrated with the base class QEWidget which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.80.2 Member Enumeration Documentation

9.80.2.1 enum QELabel::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.80.2.2 enum QELabel::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.80.2.3 enum QELabel::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.80.2.4 enum QELabel::UpdateOptions

User friendly enumerations for updateOption property - refer to QELabel::updateOptions for details.

Enumerator:

Text Data updates will update the label text.

Picture Data updates will update the label icon.

9.80.2.5 enum QELabel::updateOptions

Options for updating the label. The formatted text is used to update the label text, or select a background pixmap.

Enumerator:

UPDATE_TEXT Update the label text.

UPDATE_PIXMAP Update the label background pixmap.

9.80.2.6 enum QELabel::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.80.3 Constructor & Destructor Documentation

9.80.3.1 QELabel::QELabel (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.80.3.2 QELabel::QELabel (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.80.4 Member Function Documentation

9.80.4.1 void QELabel::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.80.5 Property Documentation

9.80.5.1 bool QELabel::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.80.5.2 bool QELabel::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.80.5.3 ArrayActions QELabel::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.80.5.4 bool QELabel::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.80.5.5 Formats QELabel::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.80.5.6 unsigned QELabel::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.80.5.7 bool QELabel::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.80.5.8 QString QELabel::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.80.5.9 Notations QELabel::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.80.5.10 QPixmap QELabel::pixmap0 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 0.

9.80.5.11 QPixmap QELabel::pixmap1 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 1.

9.80.5.12 QPixmap QELabel::pixmap2 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 2.

9.80.5.13 QPixmap QELabel::pixmap3 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 3.

9.80.5.14 QPixmap QELabel::pixmap4 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 4.

9.80.5.15 QPixmap QELabel::pixmap5 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 5.

9.80.5.16 QPixmap QELabel::pixmap6 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 6.

9.80.5.17 QPixmap QELabel::pixmap7 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 7.

9.80.5.18 int QELabel::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.80.5.19 bool QELabel::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.80.5.20 UpdateOptions QELabel::updateOption [read, write]

Determines if data updates the label text, or the label pixmap. For both options all normal string formatting is applied. If Text, the formatted text is simply presented as the label text. If Picture, the FORMATTED text is then interpreted as an integer and used to select one of the pixmaps specified by properties pixmap0 through to pixmap7.

9.80.5.21 bool QELabel::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.80.5.22 UserLevels QELabel::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.80.5.23 QString QELabel::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.5.24 QString QELabel::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.5.25 QString QELabel::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.5.26 UserLevels QELabel::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.80.5.27 QString QELabel::variable [read, write]

EPICS variable name (CA PV)

9.80.5.28 bool QELabel::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.80.5.29 QString QELabel::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.80.5.30 bool QELabel::visible [read, write]

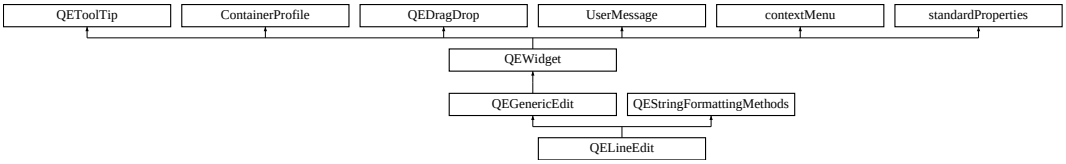
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QELabel/QELabel.h
- /tmp/epicsqt/trunk/framework/widgets/QELabel/QELabel.cpp

9.81 QELineEdit Class Reference

Inheritance diagram for QELineEdit:



Public Types

- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_-FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInteger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_-LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_-AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }

Signals

- void dbValueChanged (const QString &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.
- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.
- QELineEdit (QWidget ∗parent=0)
- QELineEdit (const QString &variableName, QWidget ∗parent=0)

Properties

- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- unsigned int
- Notations notation
- ArrayActions arrayAction

9.81.1 Member Enumeration Documentation

9.81.1.1 enum QELineEdit::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.81.1.2 enum QELineEdit::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.81.1.3 enum QELineEdit::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.81.2 Constructor & Destructor Documentation

9.81.2.1 QELineEdit::QELineEdit (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.81.2.2 QELineEdit::QELineEdit (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.81.3 Member Function Documentation

9.81.3.1 void QELineEdit::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.81.4 Property Documentation

9.81.4.1 bool QELineEdit::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.81.4.2 ArrayActions QELineEdit::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.81.4.3 Formats QELineEdit::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.81.4.4 unsigned QELineEdit::int [read, write]

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

Reimplemented from QEGenericEdit.

9.81.4.5 bool QELineEdit::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.81.4.6 QString QELineEdit::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.81.4.7 Notations QELineEdit::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.81.4.8 int QELineEdit::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.81.4.9 bool QELineEdit::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.81.4.10 bool QELineEdit::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEdit.h

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEdit.cpp

9.82 QELineEditManager Class Reference

Public Member Functions

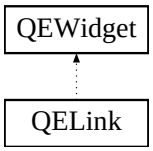
- **QELineEditManager** (QObject ∗parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **createWidget** (QWidget ∗parent)
- void **initialize** (QDesignerFormEditorInterface ∗core)

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEditManager.h

9.83 QELink Class Reference

Inheritance diagram for QELink:



Public Types

- enum **conditions** {

  CONDITION_EQ, **CONDITION_NE**, **CONDITION_GT**, **CONDITION_GE**,

  CONDITION_LT, **CONDITION_LE** }
- enum **ConditionNames** {

  Equal = QELink::CONDITION_EQ, **NotEqual** = QELink::CONDITION_NE, **GreaterThan** = QELink::CONDITION_GT, **GreaterThanOrEqual** = QELink::CONDITION_GE,

  LessThan = QELink::CONDITION_LT, **LessThanOrEqual** = QELink::CONDITION_-LE }

Public Slots

- void **in** (const bool &in)
- void **in** (const qlonglong &in)
- void **in** (const double &in)
- void **in** (const QString &in)
- void **autoFillBackground** (const bool &enable)

Signals

- void **out** (const bool &out)
- void **out** (const qlonglong &out)
- void **out** (const double &out)
- void **out** (const QString &out)

Public Member Functions

- **QELink** (QWidget ∗parent=0)
- void **setCondition** (conditions conditionIn)
- conditions **getCondition** ()
- void **setComparisonValue** (QString comparisonValue)
- QString **getComparisonValue** ()
- void **setSignalTrue** (bool signalTrue)
- bool **getSignalTrue** ()
- void **setSignalFalse** (bool signalFalse)
- bool **getSignalFalse** ()
- void **setOutTrueValue** (QString outTrueValue)
- QString **getOutTrueValue** ()
- void **setOutFalseValue** (QString outFalseValue)
- QString **getOutFalseValue** ()
- void **setRunVisible** (bool visibleIn)
- bool **getRunVisible** ()
- void **setConditionProperty** (ConditionNames condition)
- ConditionNames **getConditionProperty** ()

Protected Attributes

- conditions **condition**
- QVariant **comparisonValue**
- bool **signalTrue**
- bool **signalFalse**
- QVariant **outTrueValue**
- QVariant **outFalseValue**
- bool **visible**

Properties

- ConditionNames **condition**
- QString **comparisonValue**
- QString **outTrueValue**
- QString **outFalseValue**
- bool **runVisible**

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

- /tmp/epicsqt/trunk/framework/widgets/QELink/QELink.h
- /tmp/epicsqt/trunk/framework/widgets/QELink/QELink.cpp

9.84 QELocalEnumeration Class Reference

`#include <QELocalEnumeration.h>`

Classes

- class **localEnumerationItem**

Public Member Functions

- QELocalEnumeration ()
- QELocalEnumeration (const QString &localEnumeration)
- void setLocalEnumeration (const QString &localEnumeration)
- QString getLocalEnumeration ()
- bool isDefined ()
- QString valueToText (const QVariant &value, bool &match)
- QVariant textToValue (const QString &text, bool &ok)
- int textToInt (const QString &text, bool &ok)
- double textToDouble (const QString &text, bool &ok)

9.84.1 Detailed Description

This class allows a user defined two-way value to enumeration map. The map is define using a single string, typically a widget property string. This may then be used to replace the enumeration values provided by EPICS and/or provide an enueration set of more that 16 values. See setLocalEnumeration() for the use of 'localEnumeration'.

This functionality that this class provided was formerly embedded within QEStringFormatting.

9.84.2 Constructor & Destructor Documentation

9.84.2.1 QELocalEnumeration::QELocalEnumeration ( )

Constructors

9.84.2.2 QELocalEnumeration::QELocalEnumeration (const QString & *localEnumeration*)

Constructor with localEnumeration

9.84.3 Member Function Documentation

9.84.3.1 QString QELocalEnumeration::getLocalEnumeration ( )

Get the local enumeration strings. See setLocalEnumeration() for the use of 'localEnumeration'.

9.84.3.2 bool QELocalEnumeration::isDefined ( )

Evaluates: getLocalEnumeration.count() $>$ 0

9.84.3.3 void QELocalEnumeration::setLocalEnumeration (const QString & *localEnumeration*)

Parse the local enumeration string.

Format is:

[[$<$|$<$=|=|!=|$>$=|$>$]value1|$*$] : string1 , [[$<$|$<$=|=|!=|$>$=|$>$]value2|$*$] : string2 , [[$<$|$<$=|=|!=|$>$=|$>$]value3|$*$] : string3 , ...

Where: $<$ Less than $<$= Less than or equal = Equal (default if no operator specified) $>$= Greather than or equal $>$ Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" $<$2:"Value is less than two", =2:"Value is equal to two", $>$2:"Value is grater than 2" 3:"Beamline Available", $*$:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

Will completely re-initialises the object.

9.84.3.4 double QELocalEnumeration::textToDouble (const QString & *text,* bool & *ok*)

Generate a double value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't be formatted then 0.0 is returned and 'ok' is false.

9.84.3.5 int QELocalEnumeration::textToInt (const QString & *text,* bool & *ok*)

Generate an integer value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't be formatted then 0 is returned and 'ok' is false.

9.84.3.6 QVariant QELocalEnumeration::textToValue (const QString & *text,* bool & *ok*)

Generate a value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't be formatted an error string is returned and 'ok' is false

9.84.3.7 QString QELocalEnumeration::valueToText (const QVariant & *value,* bool & *match*)

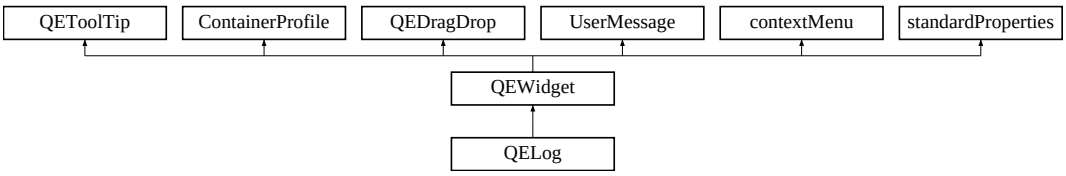
Format a variant value using local enumeration list. If the value is numeric, then the value is compared to the numeric interpretation of the enumeration values, if the value is textual, then the value is compared to the textual enumeration values.

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

- /tmp/epicsqt/trunk/framework/data/include/QELocalEnumeration.h
- /tmp/epicsqt/trunk/framework/data/src/QELocalEnumeration.cpp

9.85 QELog Class Reference

Inheritance diagram for QELog:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **MessageFilterOptions** { **Any** = UserMessage::MESSAGE_FILTER_ANY, **Match** = UserMessage::MESSAGE_FILTER_MATCH, **None** = UserMessage::MESSAGE_-FILTER_NONE }

Public Member Functions

- **QELog** (QWidget ∗pParent=0)
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnType** (bool pValue)
- bool **getShowColumnType** ()
- void **setShowColumnMessage** (bool pValue)
- bool **getShowColumnMessage** ()
- void **setShowMessageFilter** (bool pValue)
- bool **getShowMessageFilter** ()
- void **setShowClear** (bool pValue)
- bool **getShowClear** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setScrollToBottom** (bool pValue)
- bool **getScrollToBottom** ()
- void **setInfoColor** (QColor pValue)
- QColor **getInfoColor** ()
- void **setWarningColor** (QColor pValue)
- QColor **getWarningColor** ()
- void **setErrorColor** (QColor pValue)
- QColor **getErrorColor** ()
- void **clearLog** ()
- void **addLog** (int pType, QString pMessage)
- void **refreshLog** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()

- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Attributes

- _QTableWidgetLog * **qTableWidgetLog**
- QCheckBox * **qCheckBoxInfoMessage**
- QCheckBox * **qCheckBoxWarningMessage**
- QCheckBox * **qCheckBoxErrorMessage**
- QPushButton * **qPushButtonClear**
- QPushButton * **qPushButtonSave**
- QColor **qColorInfo**
- QColor **qColorWarning**
- QColor **qColorError**
- bool **scrollToBottom**
- int **detailsLayout**

Properties

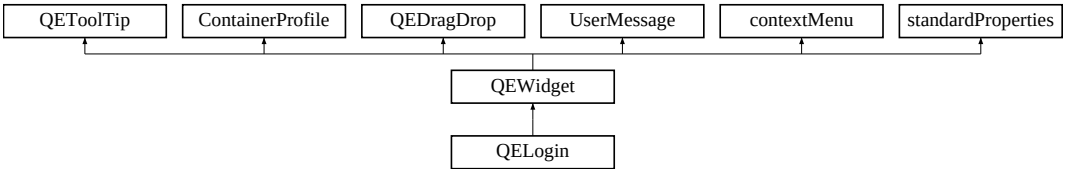
- bool **showColumnTime**
- bool **showColumnType**
- bool **showColumnMessage**
- bool **showMessageFilter**
- bool **showClear**
- bool **showSave**
- detailsLayoutProperty **detailsLayout**
- QColor **infoColor**
- QColor **warningColor**
- QColor **errorColor**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**
- unsigned **int**

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

- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.h
- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.cpp

9.86 QELogin Class Reference

Inheritance diagram for QELogin:



Signals

- void **login** ()

Public Member Functions

- **QELogin** (QWidget *pParent=0)
- bool **login** (userLevelTypes::userLevels level, QString password)
- QString **getPriorityUserPassword** ()
- QString **getPriorityScientistPassword** ()
- QString **getPriorityEngineerPassword** ()
- void **setUserPassword** (QString pValue)
- QString **getUserPassword** ()
- void **setScientistPassword** (QString pValue)
- QString **getScientistPassword** ()
- void **setEngineerPassword** (QString pValue)
- QString **getEngineerPassword** ()
- void **setCompactStyle** (bool compactStyle)
- bool **getCompactStyle** ()
- void **setStatusOnly** (bool statusOnlyIn)
- bool **getStatusOnly** ()
- QString **getUserTypeName** (userLevelTypes::userLevels type)

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

- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.87 QELoginDialog Class Reference

Public Member Functions

- **QELoginDialog** (QELogin *ownerIn)

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

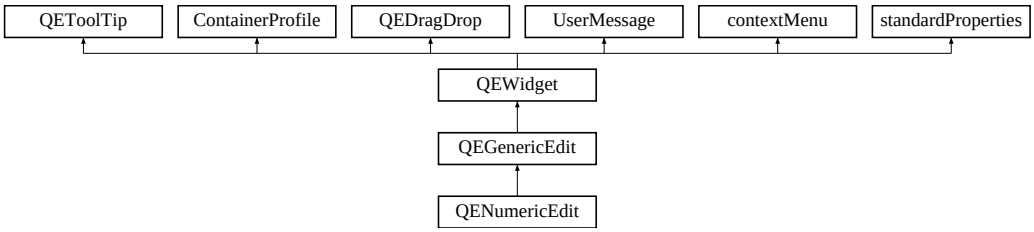
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.88 QENumericEdit Class Reference

The QENumericEdit class This class is similar to QELineEdit (both of which are derived from QLineEdit). However this class is tailored specifcially for editing numerical values.

`#include <QENumericEdit.h>`

Inheritance diagram for QENumericEdit:



Public Types

- enum Radicies { **Decimal** = 0, **Hexadecimal**, **Octal**, **Binary** }

  Specify radix, default is Decimal.
- enum Separators { **None** = 0, **Comma**, **Underscore**, **Space** }

  Specify digit 'thousands' separator character, default is none.

Signals

- void dbValueChanged (const double &out)

Public Member Functions

- QENumericEdit (QWidget *parent=0)
- QENumericEdit (const QString &variableName, QWidget *parent=0)
- virtual ~QENumericEdit ()

  Destruction.
- double **getNumericValue** ()

Protected Member Functions

- void **setAutoScale** (const bool value)
- bool **getAutoScale** ()
- void **setPropertyPrecision** (const int value)
- int **getPropertyPrecision** ()
- void **setPropertyLeadingZeros** (const int value)
- int **getPropertyLeadingZeros** ()
- void **setPropertyMinimum** (const double value)
- double **getPropertyMinimum** ()
- void **setPropertyMaximum** (const double value)
- double **getPropertyMaximum** ()
- void **setAddUnits** (bool addUnits)
- bool **getAddUnits** ()
- void **setRadix** (const Radicies value)
- Radicies **getRadix** ()
- void **setSeparator** (const Separators value)
- Separators **getSeparator** ()
- void **keyPressEvent** (QKeyEvent ∗event)
- void **focusInEvent** (QFocusEvent ∗event)
- void **mouseReleaseEvent** (QMouseEvent ∗event)
- void **establishConnection** (unsigned int variableIndex)
- qcaobject::QCaObject ∗ **createQcaItem** (unsigned int variableIndex)
- int **getPrecision** ()
- int **getLeadingZeros** ()
- double **getMinimum** ()
- double **getMaximum** ()
- int **maximumSignificance** ()
- int **getRadixValue** ()
- void setValue (const QVariant &value)

  Sets the undelying QLineEdit widget to the given value.
- QVariant getValue ()

  Gets the undelying value.
- bool writeData (const QVariant &value, QString &message)

  Write the data to the channel.

Protected Attributes

- QEFloatingFormatting **floatingFormatting**

Properties

- bool autoScale
- int precision
- int leadingZeros
- double minimum
- double maximum
- bool addUnits
- Radicies **radix**
- Separators **separator**

Friends

- class **NumericValidator**

9.88.1 Detailed Description

The QENumericEdit class This class is similar to QELineEdit (both of which are derived from QLineEdit). However this class is tailored specifcially for editing numerical values.

Note: this class based on thumb_wheel_edits.pas by same author.

9.88.2 Constructor & Destructor Documentation

9.88.2.1 QENumericEdit::QENumericEdit (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.88.2.2 QENumericEdit::QENumericEdit (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.88.3 Member Function Documentation

9.88.3.1 void QENumericEdit::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.88.4 Property Documentation

9.88.4.1 bool QENumericEdit::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.88.4.2 bool QENumericEdit::autoScale [read, write]

If true (default), display and editing of numbers using the precision, and control limits supplied with the data. If false, the precision, leadingZeros, minimum and maximum properties are used.

9.88.4.3 int QENumericEdit::leadingZeros [read, write]

Speficies the number of leading zeros. This is only used if autoScale is false. Stictly speaking, this should be an unsigned int, but designer properties editor much 'nicer' with integers.

9.88.4.4 double QENumericEdit::maximum [read, write]

Speficies the maximum allowed value. This is only used if autoScale is false.

9.88.4.5 double QENumericEdit::minimum [read, write]

Speficies the mimimum allowed value. This is only used if autoScale is false.

9.88.4.6 int QENumericEdit::precision [read, write]

Precision used for the display and editing of numbers. The default is 4. This is only used if autoScale is false. Stictly speaking, this should be an unsigned int, but designer properties editor much 'nicer' with integers.

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEdit.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEdit.cpp

9.89 QENumericEditManager Class Reference

Public Member Functions

- **QENumericEditManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const

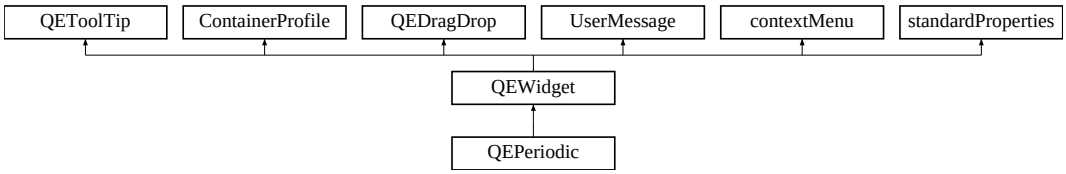
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **createWidget** (QWidget ∗parent)
- void **initialize** (QDesignerFormEditorInterface ∗core)

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEditManager.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEditManager.cpp

9.90 QEPeriodic Class Reference

Inheritance diagram for QEPeriodic:



Classes

- struct elementInfoStruct
- struct userInfoStructArray

Public Types

- enum **variableTypes** {

  VARIABLE_TYPE_NUMBER, **VARIABLE_TYPE_ATOMIC_WEIGHT**, **VARIABLE_-TYPE_MELTING_POINT**, **VARIABLE_TYPE_BOILING_POINT**,

  VARIABLE_TYPE_DENSITY, **VARIABLE_TYPE_GROUP**, **VARIABLE_TYPE_-IONIZATION_ENERGY**, **VARIABLE_TYPE_USER_VALUE_1**,

  VARIABLE_TYPE_USER_VALUE_2 }
- enum **presentationOptions** { **PRESENTATION_BUTTON_AND_LABEL**, **PRESENTATION_-BUTTON_ONLY**, **PRESENTATION_LABEL_ONLY** }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

- enum **PresentationOptions** { **buttonAndLabel** = QEPeriodic::PRESENTATION_-BUTTON_AND_LABEL, **buttonOnly** = QEPeriodic::PRESENTATION_BUTTON_-ONLY, **labelOnly** = QEPeriodic::PRESENTATION_LABEL_ONLY }
- enum **VariableTypes** {

  Number = QEPeriodic::VARIABLE_TYPE_NUMBER, **atomicWeight** = QEPeriodic::VARIABLE_-TYPE_ATOMIC_WEIGHT, **meltingPoint** = QEPeriodic::VARIABLE_TYPE_MELTING_-POINT, **boilingPoint** = QEPeriodic::VARIABLE_TYPE_BOILING_POINT,

  density = QEPeriodic::VARIABLE_TYPE_DENSITY, **group** = QEPeriodic::VARIABLE_-TYPE_GROUP, **ionizationEnergy** = QEPeriodic::VARIABLE_TYPE_IONIZATION_-ENERGY, **userValue1** = QEPeriodic::VARIABLE_TYPE_USER_VALUE_1,

  userValue2 = QEPeriodic::VARIABLE_TYPE_USER_VALUE_2 }

Signals

- void dbValueChanged (const double &out)
- void dbElementChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- **QEPeriodic** (QWidget *parent=0)
- **QEPeriodic** (const QString &variableName, QWidget *parent=0)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setPresentationOption** (presentationOptions presentationOptionIn)
- presentationOptions **getPresentationOption** ()
- void **setVariableType1** (variableTypes variableType1In)
- variableTypes **getVariableType1** ()
- void **setVariableType2** (variableTypes variableType2In)
- variableTypes **getVariableType2** ()
- void **setVariableTolerance1** (double variableTolerance1In)
- double **getVariableTolerance1** ()
- void **setVariableTolerance2** (double variableTolerance2In)
- double **getVariableTolerance2** ()
- void **setUserInfo** (QString userInfo)
- QString **getUserInfo** ()
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

- void **setPresentationOptionProperty** (PresentationOptions presentationOption)
- PresentationOptions **getPresentationOptionProperty** ()
- void **setVariableType1Property** (VariableTypes variableType)
- void **setVariableType2Property** (VariableTypes variableType)
- VariableTypes **getVariableType1Property** ()
- VariableTypes **getVariableType2Property** ()

Public Attributes

- userInfoStruct **userInfo** [NUM_ELEMENTS]

Static Public Attributes

- static elementInfoStruct **elementInfo** [NUM_ELEMENTS]

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **localEnabled**
- bool allowDrop
- variableTypes **variableType1**
- variableTypes **variableType2**
- double **variableTolerance1**
- double **variableTolerance2**

Properties

- QString writeButtonVariable1
- QString writeButtonVariable2
- QString readbackLabelVariable1
- QString readbackLabelVariable2
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- PresentationOptions **presentationOption**
- VariableTypes **variableType1**
- VariableTypes **variableType2**
- QString **userInfo**

9.90.1 Member Enumeration Documentation

9.90.1.1 enum QEPeriodic::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.90.2 Member Function Documentation

9.90.2.1 void QEPeriodic::dbElementChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.90.2.2 void QEPeriodic::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.90.3 Member Data Documentation

9.90.3.1 bool QEPeriodic::allowDrop [read, write, protected]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.90.4 Property Documentation

9.90.4.1 bool QEPeriodic::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.90.4.2 unsigned QEPeriodic::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.90.4.3 QString QEPeriodic::readbackLabelVariable1 [read, write]

EPICS variable name (CA PV). This variable is used to read the value to the first of two positioners to determine which (if any) element is currently selected.

9.90.4.4 QString QEPeriodic::readbackLabelVariable2 [read, write]

EPICS variable name (CA PV). This variable is used to read the value to the second of two positioners to determine which (if any) element is currently selected.

9.90.4.5 bool QEPeriodic::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.90.4.6 UserLevels QEPeriodic::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.90.4.7 QString QEPeriodic::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.90.4.8 QString QEPeriodic::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.90.4.9 QString QEPeriodic::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.90.4.10 UserLevels QEPeriodic::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.90.4.11 bool QEPeriodic::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.90.4.12 QString QEPeriodic::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.90.4.13 bool QEPeriodic::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.90.4.14 QString QEPeriodic::writeButtonVariable1 [read, write]

EPICS variable name (CA PV). This variable is used to write a value to the first of two positioners that will position the select element.

9.90.4.15 QString QEPeriodic::writeButtonVariable2 [read, write]

EPICS variable name (CA PV). This variable is used to write a value to the second of two positioners that will position the select element.

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.cpp

9.91 QEPeriodicComponentData Class Reference

Public Attributes

- unsigned int **variableIndex1**
- double **lastData1**
- bool **haveLastData1**
- unsigned int **variableIndex2**
- double **lastData2**
- bool **haveLastData2**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.92 QEPeriodicTaskMenu Class Reference

Public Member Functions

- **QEPeriodicTaskMenu** (QEPeriodic ∗periodic, QObject ∗parent)
- QAction ∗ **preferredEditAction** () const
- QList< QAction ∗ > **taskActions** () const

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenuExtension.cpp

9.93 QEPeriodicTaskMenuFactory Class Reference

Public Member Functions

- **QEPeriodicTaskMenuFactory** (QExtensionManager ∗parent=0)

Protected Member Functions

- QObject ∗ **createExtension** (QObject ∗object, const QString &iid, QObject ∗parent) const

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenuExtension.cpp

9.94 QEpicsPV Class Reference

Public Slots

- const QVariant & **set** (QVariant value, int delay=-1)
- void **setPV** (const QString &_pvName="")

Signals

- void **connectionChanged** (bool connected)
- void **connected** ()
- void **disconnected** ()
- void **valueChanged** (const QVariant &value)
- void **valueUpdated** (const QVariant &value)
- void **valueInited** (const QVariant &value)

Public Member Functions

- **QEpicsPV** (const QString &_pvName, QObject ∗parent=0)
- **QEpicsPV** (QObject ∗parent=0)
- const QVariant & **get** () const
- void **needUpdated** () const
- const QVariant & **getUpdated** (int delay=defaultDelay) const
- bool **isConnected** () const
- const QStringList & **getEnum** () const
- const QString & **pv** () const
- const QVariant & **getReady** (int delay=defaultDelay) const

Static Public Member Functions

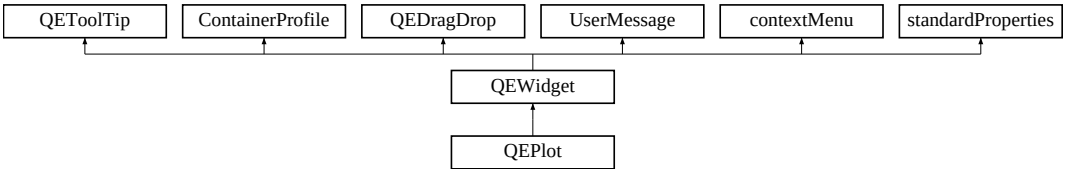
- static void **setDebugLevel** (unsigned level=0)
- static QVariant **get** (const QString &_pvName, int delay=defaultDelay)
- static QVariant **set** (QString &_pvName, const QVariant &value, int delay=-1)

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

- /tmp/epicsqt/trunk/framework/data/include/qepicspv.h
- /tmp/epicsqt/trunk/framework/data/src/qepicspv.cpp

9.95 QEPlot Class Reference

Inheritance diagram for QEPlot:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum **TraceStyles** { **Lines** = QwtPlotCurve::Lines, **Sticks** = QwtPlotCurve::Sticks, **Steps** = QwtPlotCurve::Steps, **Dots** = QwtPlotCurve::Dots }

Signals

- void dbValueChanged (const double &out)
- void dbValueChanged (const QVector< double > &out)

Public Member Functions

- **QEPlot** (QWidget *parent=0)
- **QEPlot** (const QString &variableName, QWidget *parent=0)
- void **setYMin** (double yMin)
- double **getYMin** ()
- void **setYMax** (double yMax)
- double **getYMax** ()
- void **setAutoScale** (bool autoScale)
- bool **getAutoScale** ()
- void **setAxisEnableX** (bool axisEnableXIn)
- bool **getAxisEnableX** ()
- void **setAxisEnableY** (bool axisEnableYIn)
- bool **getAxisEnableY** ()
- QString **getTitle** ()
- void **setBackgroundColor** (QColor backgroundColor)
- QColor **getBackgroundColor** ()
- void **setTraceStyle** (QwtPlotCurve::CurveStyle traceStyle, const unsigned int variableIndex)
- QwtPlotCurve::CurveStyle **getTraceStyle** (const unsigned int variableIndex)
- void **setTraceColor** (QColor traceColor, const unsigned int variableIndex)
- void **setTraceColor1** (QColor traceColor)

- void **setTraceColor2** (QColor traceColor)
- void **setTraceColor3** (QColor traceColor)
- void **setTraceColor4** (QColor traceColor)
- QColor **getTraceColor** (const unsigned int variableIndex)
- QColor **getTraceColor1** ()
- QColor **getTraceColor2** ()
- QColor **getTraceColor3** ()
- QColor **getTraceColor4** ()
- void **setTraceLegend1** (QString traceLegend)
- void **setTraceLegend2** (QString traceLegend)
- void **setTraceLegend3** (QString traceLegend)
- void **setTraceLegend4** (QString traceLegend)
- QString **getTraceLegend1** ()
- QString **getTraceLegend2** ()
- QString **getTraceLegend3** ()
- QString **getTraceLegend4** ()
- void **setXUnit** (QString xUnit)
- QString **getXUnit** ()
- void **setYUnit** (QString yUnit)
- QString **getYUnit** ()
- void **setGridEnableMajorX** (bool gridEnableMajorXIn)
- void **setGridEnableMajorY** (bool gridEnableMajorYIn)
- void **setGridEnableMinorX** (bool gridEnableMinorXIn)
- void **setGridEnableMinorY** (bool gridEnableMinorYIn)
- bool **getGridEnableMajorX** ()
- bool **getGridEnableMajorY** ()
- bool **getGridEnableMinorX** ()
- bool **getGridEnableMinorY** ()
- void **setGridMajorColor** (QColor gridMajorColorIn)
- void **setGridMinorColor** (QColor gridMinorColorIn)
- QColor **getGridMajorColor** ()
- QColor **getGridMinorColor** ()
- void **setXStart** (double xStart)
- double **getXStart** ()
- void **setXIncrement** (double xIncrement)
- double **getXIncrement** ()
- void **setTimeSpan** (unsigned int timeSpan)
- unsigned int **getTimeSpan** ()
- void **setTickRate** (unsigned int tickRate)
- unsigned int **getTickRate** ()
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.

- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void **setTraceStyle1** (TraceStyles traceStyle)
- void **setTraceStyle2** (TraceStyles traceStyle)
- void **setTraceStyle3** (TraceStyles traceStyle)
- void **setTraceStyle4** (TraceStyles traceStyle)
- TraceStyles **getTraceStyle1** ()
- TraceStyles **getTraceStyle2** ()
- TraceStyles **getTraceStyle3** ()
- TraceStyles **getTraceStyle4** ()

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **localEnabled**
- bool allowDrop

Properties

- QString variable1
- QString variable2
- QString variable3
- QString variable4
- QString variableSubstitutions
- bool variableAsToolTip
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

- bool displayAlarmState
- QColor **traceColor1**
- QColor **traceColor2**
- QColor **traceColor3**
- QColor **traceColor4**
- TraceStyles **traceStyle1**
- TraceStyles **traceStyle2**
- TraceStyles **traceStyle3**
- TraceStyles **traceStyle4**
- QString **traceLegend1**
- QString **traceLegend2**
- QString **traceLegend3**
- QString **traceLegend4**
- QString **title**
- QColor **backgroundColor**
- QString **xUnit**
- QString **yUnit**

9.95.1 Member Enumeration Documentation

9.95.1.1 enum QEPlot::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.95.2 Member Function Documentation

9.95.2.1 void QEPlot::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.95.2.2 void QEPlot::dbValueChanged (const QVector< double > & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.95.3 Member Data Documentation

9.95.3.1 bool QEPlot::allowDrop [read, write, protected]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.95.4 Property Documentation

9.95.4.1 bool QEPlot::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.95.4.2 unsigned QEPlot::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.95.4.3 UserLevels QEPlot::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.95.4.4 QString QEPlot::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.95.4.5 QString QEPlot::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.95.4.6 QString QEPlot::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.95.4.7 UserLevels QEPlot::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.95.4.8 QString QEPlot::variable1 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the first trace.

9.95.4.9 QString QEPlot::variable2 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the second trace.

9.95.4.10 QString QEPlot::variable3 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the third trace.

9.95.4.11 QString QEPlot::variable4 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the fourth trace.

9.95.4.12 bool QEPlot::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.95.4.13 QString QEPlot::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.95.4.14 bool QEPlot::visible [read, write]

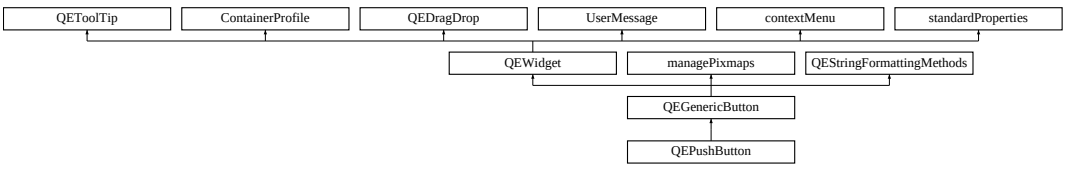
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.h
- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.cpp

9.96 QEPushButton Class Reference

Inheritance diagram for QEPushButton:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_-FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInte-ger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_-LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_-AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QEGenericButton::UPDATE_TEXT, Icon = QEGenericButton::UPDATE_-ICON, TextAndIcon = QEGenericButton::UPDATE_TEXT_AND_ICON, State = QEGenericButton::UPDATE_STATE }

  User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.
- enum CreationOptionNames { Open = QEForm::CREATION_OPTION_OPEN, NewTab = QEForm::CREATION_OPTION_NEW_TAB, NewWindow = QEForm::CREATION_-OPTION_NEW_WINDOW }

  Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Public Slots

- void launchGui (QString guiName, QEForm::creationOptions creationOption)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.
- void newGui (QString guiName, QEForm::creationOptions creationOption)

  Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.
- void pressed (int value)
- void released (int value)
- void clicked (int value)

Public Member Functions

- QEPushButton (QWidget ∗parent=0)
- QEPushButton (const QString &variableName, QWidget ∗parent=0)
- UserLevels getUserLevelVisibilityProperty ()

Access function for userLevelVisibility property - refer to userLevelVisibility property for details.

- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.
- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.

Properties

- QString variable
- QString altReadbackVariable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits

- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- Qt::Alignment alignment
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7
- QString password
- bool confirmAction
- bool writeOnPress
- bool writeOnRelease
- bool writeOnClick
- QString pressText
- QString releaseText
- QString clickText
- QString clickCheckedText
- QString labelText
- QString program
- QStringList arguments
- QString guiFile
- CreationOptionNames creationOption
- QString prioritySubstitutions

9.96.1 Member Enumeration Documentation

9.96.1.1 enum QEPushButton::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.96.1.2 enum QEPushButton::CreationOptionNames

Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Enumerator:

Open Replace the current GUI with the new GUI.

NewTab Open new GUI in a new tab.

NewWindow Open new GUI in a new window.

9.96.1.3 enum QEPushButton::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.96.1.4 enum QEPushButton::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.96.1.5 enum QEPushButton::UpdateOptions

User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.

Enumerator:

Text Data updates will update the button text.

Icon Data updates will update the button icon.

TextAndIcon Data updates will update the button text and icon.

State Data updates will update the button state (checked or unchecked)

9.96.1.6 enum QEPushButton::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and #userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.96.2 Constructor & Destructor Documentation

9.96.2.1 QEPushButton::QEPushButton (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.96.2.2 QEPushButton::QEPushButton (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.96.3 Member Function Documentation

9.96.3.1 void QEPushButton::clicked (int *value*) [signal]

Button has been Clicked. The value emitted is the integer interpretation of the clickText property (or the clickCheckedText property if the button was checked)

9.96.3.2 void QEPushButton::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.96.3.3 void QEPushButton::launchGui (QString *guiName*, QEForm::creationOptions *creationOption*) [inline, slot]

Default slot used to create a new GUI if there is no slot indicated in the ContainerProfile class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not

specify a slot to use for creating new windows (through the ContainerProfile class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the ContainerProfile class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from QEGenericButton.

9.96.3.4 void QEPushButton::pressed (int *value*) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

9.96.3.5 void QEPushButton::released (int *value*) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

9.96.4 Property Documentation

9.96.4.1 bool QEPushButton::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.96.4.2 Qt::Alignment QEPushButton::alignment [read, write]

Set the buttons text alignment. Left justification is particularly useful when displaying quickly changing numeric data updates.

9.96.4.3 bool QEPushButton::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.96.4.4 QString QEPushButton::altReadbackVariable [read, write]

EPICS variable name (CA PV). This variable is used to provide a readback value when different to the variable written to by a button press.

9.96.4.5 QStringList QEPushButton::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from QEGenericButton.

9.96.4.6 ArrayActions QEPushButton::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.96.4.7 QString QEPushButton::clickCheckedText [read, write]

Text used to compare with text written or read to determine if push button should be marked as checked. Note, must be an exact match following formatting of data updates. When writing values, the 'pressText', 'ReleaseText', or 'clickedtext' must match this property to cause the button to be checked when the write occurs.

Good example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is 'On'. In this example, the push button will be checked when a data update occurs with a value of 1 or when the button is clicked.

Bad example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is '1'. In this example, the push button will be checked when a data update occurs with a value of 1 but, although a valid value will be written when clicked, the button will not be checked when clicked as '1' is not the same as 'On'.

Reimplemented from QEGenericButton.

9.96.4.8 QString QEPushButton::clickText [read, write]

Value written when user clicks button if 'writeOnClick' property is true

Reimplemented from QEGenericButton.

9.96.4.9 bool QEPushButton::confirmAction [read, write]

If true, a dialog will be presented asking the user to confirm if the button action should be carried out

9.96.4.10 CreationOptionNames QEPushButton::creationOption [read, write]

Creation options when opening a new GUI. Open a new window, open a new tab, or replace the current window. the creation option is supplied when the button generates a newGui signal. Application code connected to this signal should honour this request if possible. When used within the QEGui application, the QEGui application creates a new window, new tab, or replaces the current window as appropriate.

Reimplemented from QEGenericButton.

9.96.4.11 bool QEPushButton::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.96.4.12 Formats QEPushButton::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.96.4.13 QString QEPushButton::guiFile [read, write]

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QEform in which the QEPushButton is located, relative to the any path in the path list published in the ContainerProfile class, or relative to the current path. See QEWidget::openQEFile() in QEWidget.cpp for details.

9.96.4.14 unsigned QEPushButton::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.96.4.15 QString QEPushButton::labelText [read, write]

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! For example, a button in a sub form may have a 'labelText' property of 'Turn Pump On'. When the sub form is used twice in a main form with substitutions PUMPNUM=1 and PUMPNUM=2 respectively, the two identical buttons in the sub forms will have the labels 'Turn Pump 1 On' and 'Turn Pump 2 On' respectively.

Reimplemented from QEGenericButton.

9.96.4.16 bool QEPushButton::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.96.4.17 QString QEPushButton::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.96.4.18 Notations QEPushButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.96.4.19 QString QEPushButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.96.4.20 QPixmap QEPushButton::pixmap0 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 0

9.96.4.21 QPixmap QEPushButton::pixmap1 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 1

9.96.4.22 QPixmap QEPushButton::pixmap2 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 2

9.96.4.23 QPixmap QEPushButton::pixmap3 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 3

9.96.4.24 QPixmap QEPushButton::pixmap4 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 4

9.96.4.25 QPixmap QEPushButton::pixmap5 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 5

9.96.4.26 QPixmap QEPushButton::pixmap6 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 6

9.96.4.27 QPixmap QEPushButton::pixmap7 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 7

9.96.4.28 int QEPushButton::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.96.4.29 QString QEPushButton::pressText [read, write]

Value written when user presses button if 'writeOnPress' property is true

Reimplemented from QEGenericButton.

9.96.4.30 QString QEPushButton::prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly usefull when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substittions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from QEGenericButton.

9.96.4.31 QString QEPushButton::program [read, write]

Program to run when the button is clicked. No attempt to run a program is made if this property is empty. Example: firefox

Reimplemented from QEGenericButton.

9.96.4.32 QString QEPushButton::releaseText [read, write]

Value written when user releases button if 'writeOnRelease' property is true

Reimplemented from QEGenericButton.

9.96.4.33 bool QEPushButton::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.96.4.34 bool QEPushButton::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.96.4.35 UpdateOptions QEPushButton::updateOption [read, write]

Update options (text, pixmap, both, or state (checked or unchecked)

Reimplemented from QEGenericButton.

9.96.4.36 bool QEPushButton::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.96.4.37 UserLevels QEPushButton::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.96.4.38 QString QEPushButton::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.96.4.39 QString QEPushButton::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.96.4.40 QString QEPushButton::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.96.4.41 UserLevels QEPushButton::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.96.4.42 QString QEPushButton::variable [read, write]

EPICS variable name (CA PV). This variable is used for both writing (on button press), and reading if subscribed and no alternate readback variable is provided.

9.96.4.43 bool QEPushButton::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.96.4.44 QString QEPushButton::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.96.4.45 bool QEPushButton::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.96.4.46 bool QEPushButton::writeOnClick [read, write]

If true, the 'clickText' property is written when the button is clicked. Default is true

Reimplemented from QEGenericButton.

9.96.4.47 bool QEPushButton::writeOnPress [read, write]

If true, the 'pressText' property is written when the button is pressed. Default is false

Reimplemented from QEGenericButton.

9.96.4.48 bool QEPushButton::writeOnRelease [read, write]

If true, the 'releaseText' property is written when the button is released. Default is false

Reimplemented from QEGenericButton.

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEPushButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEPushButton.cpp

9.97 QEPVNameLists Class Reference

Public Member Functions

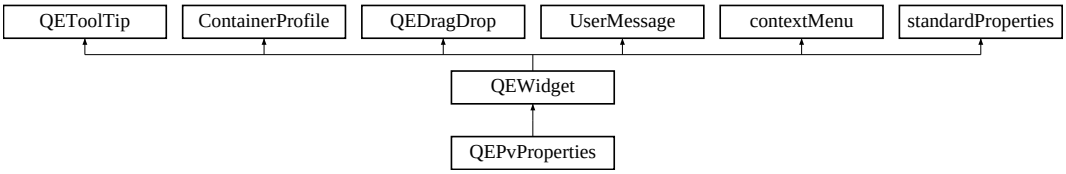
- void **prependOrMoveToFirst** (const QString &item)
- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.98 QEPvProperties Class Reference

Inheritance diagram for QEPvProperties:



Classes

- class OwnWidgets

Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void **setCurrentBoxIndex** (int index)

Public Member Functions

- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- **QEPvProperties** (QWidget *parent=0)
- **QEPvProperties** (const QString &variableName, QWidget *parent=0)
- QSize **sizeHint** () const

Protected Member Functions

- void **resizeEvent** (QResizeEvent *event)
- void **establishConnection** (unsigned int variableIndex)
- void scaleBy (const int m, const int d)
- qcaobject::QCaObject * **createQcaItem** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **mousePressEvent** (QMouseEvent *event)
- void saveConfiguration (PersistanceManager *pm)
- void restoreConfiguration (PersistanceManager *pm, restorePhases restorePhase)
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.98.1 Member Enumeration Documentation

9.98.1.1 enum QEPvProperties::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.98.2 Member Function Documentation

9.98.2.1 void QEPvProperties::restoreConfiguration (PersistanceManager * , restorePhases) [protected, virtual]

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the PersistanceManager. For example, a QEStripChart may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented from QEWidget.

9.98.2.2 void QEPvProperties::saveConfiguration (PersistanceManager *) [protected, virtual]

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the PersistanceManager. For example, a QEStripChart

may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented from QEWidget.

9.98.2.3 void QEPvProperties::scaleBy (const int , const int) [protected, virtual]

Any QEWidget that requires additional scaling, i.e. above and beyond the standard scaling applied to size, minimum size, maximum size and font size, may override this function in order to perform any bespoke scaling need by the widget (for example see QEShape). The scaling is defined using a rational number specifed by two integers (m, d). The first (m) parameter is the multiplier and the second (d) parameter is the divisor. For example, if m = 4 and d = 5, then an 80% scaling should be applied. And if m = 5 and d = 4, and a 125% scaling is required.

Reimplemented from QEWidget.

9.98.3 Property Documentation

9.98.3.1 bool QEPvProperties::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.98.3.2 bool QEPvProperties::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.98.3.3 unsigned QEPvProperties::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.98.3.4 UserLevels QEPvProperties::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user

mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.98.3.5 QString QEPvProperties::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.98.3.6 QString QEPvProperties::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.98.3.7 QString QEPvProperties::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.98.3.8 UserLevels QEPvProperties::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.98.3.9 QString QEPvProperties::variable [read, write]

EPICS variable name (CA PV)

9.98.3.10 bool QEPvProperties::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.98.3.11 QString QEPvProperties::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.98.3.12 bool QEPvProperties::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.99 QEPvPropertiesManager Class Reference

Public Member Functions

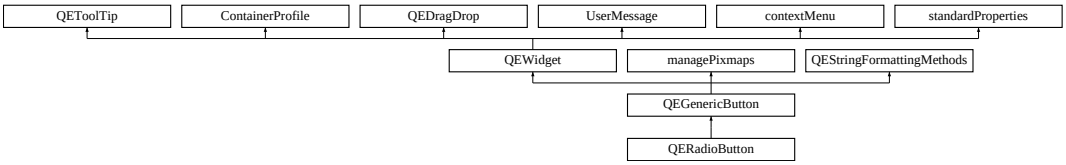
- **QEPvPropertiesManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget * **createWidget** (QWidget *parent)
- void **initialize** (QDesignerFormEditorInterface *core)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesManager.cpp

9.100 QERadioButton Class Reference

Inheritance diagram for QERadioButton:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum Formats {

  Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_-FLOATING, Integer = QEStringFormatting::FORMAT_INTEGER, UnsignedInte-ger = QEStringFormatting::FORMAT_UNSIGNEDINTEGER,

  Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_-LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEStringFormatting::NOTATION_-AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QEGenericButton::UPDATE_TEXT, Icon = QEGenericButton::UPDATE_-ICON, TextAndIcon = QEGenericButton::UPDATE_TEXT_AND_ICON, State = QEGenericButton::UPDATE_STATE }

  User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.
- enum CreationOptionNames { Open = QEForm::CREATION_OPTION_OPEN, NewTab = QEForm::CREATION_OPTION_NEW_TAB, NewWindow = QEForm::CREATION_-OPTION_NEW_WINDOW }

  Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Public Slots

- void launchGui (QString guiName, QEForm::creationOptions creationOption)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

- void newGui (QString guiName, QEForm::creationOptions creationOption)

  Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.
- void pressed (int value)
- void released (int value)
- void clicked (int value)

Public Member Functions

- QERadioButton (QWidget *parent=0)
- QERadioButton (const QString &variableName, QWidget *parent=0)
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setFormatProperty (Formats format)

  Access function for format property - refer to format property for details.
- Formats getFormatProperty ()

  Access function for format property - refer to format property for details.
- void setNotationProperty (Notations notation)

  Access function for notation property - refer to notation property for details.
- Notations getNotationProperty ()

  Access function for notation property - refer to notation property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for arrayAction property - refer to arrayAction property for details.
- ArrayActions getArrayActionProperty ()

  Access function for arrayAction property - refer to arrayAction property for details.

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip

- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- Qt::Alignment alignment
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7
- QString password
- bool confirmAction
- bool writeOnPress
- bool writeOnRelease
- bool writeOnClick
- QString pressText
- QString releaseText
- QString clickText
- QString clickCheckedText
- QString labelText
- QString program
- QStringList arguments
- QString guiFile
- CreationOptionNames creationOption
- QString prioritySubstitutions

9.100.1 Member Enumeration Documentation

9.100.1.1 enum QERadioButton::ArrayActions

User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.100.1.2 enum QERadioButton::CreationOptionNames

Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Enumerator:

Open Replace the current GUI with the new GUI.

NewTab Open new GUI in a new tab.

NewWindow Open new GUI in a new window.

9.100.1.3 enum QERadioButton::Formats

User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.100.1.4 enum QERadioButton::Notations

User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.100.1.5 enum QERadioButton::UpdateOptions

User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.

Enumerator:

Text Data updates will update the button text.

Icon Data updates will update the button icon.

TextAndIcon Data updates will update the button text and icon.

State Data updates will update the button state (checked or unchecked)

9.100.1.6 enum QERadioButton::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.100.2 Constructor & Destructor Documentation

9.100.2.1 QERadioButton::QERadioButton (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.100.2.2 QERadioButton::QERadioButton (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.100.3 Member Function Documentation

9.100.3.1 void QERadioButton::clicked (int *value*) [signal]

Button has been Clicked. The value emitted is the integer interpretation of the clickText property (or the clickCheckedText property if the button was checked)

9.100.3.2 void QERadioButton::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.100.3.3 void QERadioButton::launchGui (QString *guiName,* QEForm::creationOptions *creationOption*) [inline, slot]

Default slot used to create a new GUI if there is no slot indicated in the ContainerProfile class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not specify a slot to use for creating new windows (through the ContainerProfile class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the ContainerProfile class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from QEGenericButton.

9.100.3.4 void QERadioButton::pressed (int *value*) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

9.100.3.5 void QERadioButton::released (int *value*) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

9.100.4 Property Documentation

9.100.4.1 bool QERadioButton::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.100.4.2 Qt::Alignment QERadioButton::alignment [read, write]

Set the buttons text alignment. Left justification is particularly useful when displaying quickly changing numeric data updates.

9.100.4.3 bool QERadioButton::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.100.4.4 QStringList QERadioButton::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from QEGenericButton.

9.100.4.5 ArrayActions QERadioButton::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.100.4.6 QString QERadioButton::clickCheckedText [read, write]

Text used to compare with text written or read to determine if push button should be marked as checked. Note, must be an exact match following formatting of data updates. When writing values, the 'pressText', 'ReleaseText', or 'clickedtext' must match this property to cause the button to be checked when the write occurs.

Good example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is 'On'. In this example, the push button will be checked when a data update occurs with a value of 1 or when the button is clicked.

Bad example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is '1'. In this example, the push button will be checked when a data update occurs with a value of 1 but, although a valid value will be written when clicked, the button will not be checked when clicked as '1' is not the same as 'On'.

Reimplemented from QEGenericButton.

9.100.4.7 QString QERadioButton::clickText [read, write]

Value written when user clicks button if 'writeOnClick' property is true

Reimplemented from QEGenericButton.

9.100.4.8 bool QERadioButton::confirmAction [read, write]

If true, a dialog will be presented asking the user to confirm if the button action should be carried out

9.100.4.9 CreationOptionNames QERadioButton::creationOption [read, write]

Creation options when opening a new GUI. Open a new window, open a new tab, or replace the current window. the creation option is supplied when the button generates a newGui signal. Application code connected to this signal should honour this request if possible. When used within the QEGui application, the QEGui application creates a new window, new tab, or replaces the current window as appropriate.

Reimplemented from QEGenericButton.

9.100.4.10 bool QERadioButton::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.100.4.11 Formats QERadioButton::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.100.4.12 QString QERadioButton::guiFile [read, write]

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QEform in which the QEPushButton is located, relative to the any path in the path list published in the ContainerProfile class, or relative to the current path. See QEWidget::openQEFile() in QEWidget.cpp for details.

9.100.4.13 unsigned QERadioButton::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.100.4.14 QString QERadioButton::labelText [read, write]

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! For example, a button in a sub form may have a 'labelText' property of 'Turn Pump On'. When the sub form is used twice in a main form with substitutions PUMPNUM=1 and PUMPNUM=2 respectively, the two identical buttons in the sub forms will have the labels 'Turn Pump 1 On' and 'Turn Pump 2 On' respectively.

Reimplemented from QEGenericButton.

9.100.4.15 bool QERadioButton::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.100.4.16 QString QERadioButton::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.100.4.17 Notations QERadioButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.100.4.18 QString QERadioButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.100.4.19 QPixmap QERadioButton::pixmap0 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 0

9.100.4.20 QPixmap QERadioButton::pixmap1 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 1

9.100.4.21 QPixmap QERadioButton::pixmap2 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 2

9.100.4.22 QPixmap QERadioButton::pixmap3 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 3

9.100.4.23 QPixmap QERadioButton::pixmap4 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 4

9.100.4.24 QPixmap QERadioButton::pixmap5 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 5

9.100.4.25 QPixmap QERadioButton::pixmap6 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 6

9.100.4.26 QPixmap QERadioButton::pixmap7 [read, write]

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 7

9.100.4.27 int QERadioButton::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.100.4.28 QString QERadioButton::pressText [read, write]

Value written when user presses button if 'writeOnPress' property is true

Reimplemented from QEGenericButton.

9.100.4.29 QString QERadioButton::prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly usefull when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substittions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from QEGenericButton.

9.100.4.30 QString QERadioButton::program [read, write]

Program to run when the button is clicked. No attempt to run a program is made if this property is empty. Example: firefox

Reimplemented from QEGenericButton.

9.100.4.31 QString QERadioButton::releaseText [read, write]

Value written when user releases button if 'writeOnRelease' property is true

Reimplemented from QEGenericButton.

9.100.4.32 bool QERadioButton::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.100.4.33 bool QERadioButton::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.100.4.34 UpdateOptions QERadioButton::updateOption [read, write]

Update options (text, pixmap, both, or state (checked or unchecked)

Reimplemented from QEGenericButton.

9.100.4.35 bool QERadioButton::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.100.4.36 UserLevels QERadioButton::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.100.4.37 QString QERadioButton::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.100.4.38 QString QERadioButton::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager

class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.100.4.39 QString QERadioButton::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.100.4.40 UserLevels QERadioButton::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.100.4.41 QString QERadioButton::variable [read, write]

EPICS variable name (CA PV)

9.100.4.42 bool QERadioButton::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.100.4.43 QString QERadioButton::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.100.4.44 bool QERadioButton::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.100.4.45 bool QERadioButton::writeOnClick [read, write]

If true, the 'clickText' property is written when the button is clicked. Default is true

Reimplemented from QEGenericButton.

9.100.4.46 bool QERadioButton::writeOnPress [read, write]

If true, the 'pressText' property is written when the button is pressed. Default is false

Reimplemented from QEGenericButton.

9.100.4.47 bool QERadioButton::writeOnRelease [read, write]

If true, the 'releaseText' property is written when the button is released. Default is false

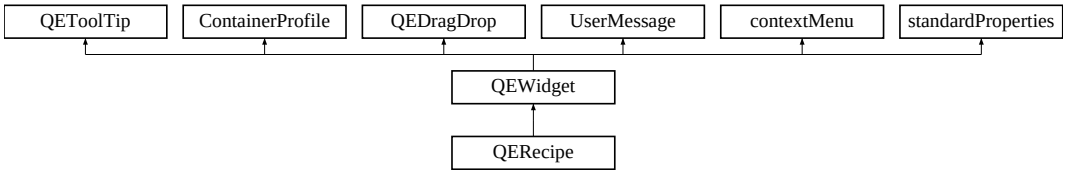
Reimplemented from QEGenericButton.

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QERadioButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QERadioButton.cpp

9.101 QERecipe Class Reference

Inheritance diagram for QERecipe:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }

Public Member Functions

- **QERecipe** (QWidget *pParent=0)

- void **setRecipeDescription** (QString pValue)
- QString **getRecipeDescription** ()
- void **setShowRecipeList** (bool pValue)
- bool **getShowRecipeList** ()
- void **setShowNew** (bool pValue)
- bool **getShowNew** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setShowDelete** (bool pValue)
- bool **getShowDelete** ()
- void **setShowApply** (bool pValue)
- bool **getShowApply** ()
- void **setShowRead** (bool pValue)
- bool **getShowRead** ()
- void **setShowFields** (bool pValue)
- bool **getShowFields** ()
- void **setConfigurationType** (int pValue)
- int **getConfigurationType** ()
- void **setConfigurationFile** (QString pValue)
- QString **getConfigurationFile** ()
- void **setRecipeFile** (QString pValue)
- QString **getRecipeFile** ()
- void **setConfigurationText** (QString pValue)
- QString **getConfigurationText** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- bool **saveRecipeList** ()
- void **refreshRecipeList** ()
- void **refreshButton** ()
- void **userLevelChanged** (userLevelTypes::userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)
- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

Protected Attributes

- QLabel * **qLabelRecipeDescription**
- QComboBox * **qComboBoxRecipeList**
- QPushButton * **qPushButtonNew**
- QPushButton * **qPushButtonSave**
- QPushButton * **qPushButtonDelete**
- QPushButton * **qPushButtonApply**
- QPushButton * **qPushButtonRead**
- QEConfiguredLayout * **qEConfiguredLayoutRecipeFields**
- QDomDocument **document**
- QString **recipeFile**
- QString **filename**
- int **detailsLayout**
- int **currentUserType**

Properties

- QString **recipeDescription**
- bool **showRecipeList**
- bool **showNew**
- bool **showSave**
- bool **showDelete**
- bool **showApply**
- bool **showRead**
- bool **showFields**
- configurationTypesProperty **configurationType**
- QString **configurationFile**
- QString **configurationText**
- detailsLayoutProperty **detailsLayout**
- userTypesProperty **currentUserType**

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

- /tmp/epicsqt/trunk/framework/widgets/QERecipe/QERecipe.h
- /tmp/epicsqt/trunk/framework/widgets/QERecipe/QERecipe.cpp

9.102 QERecordFieldName Class Reference

Static Public Member Functions

- static QString **recordName** (const QString &pvName)
- static QString **fieldName** (const QString &pvName)
- static QString **fieldPvName** (const QString &pvName, const QString &field)
- static QString **rtypePvName** (const QString &pvName)

- static bool **pvNameIsValid** (const QString &pvName)
- static bool **extractPvName** (const QString &item, QString &pvName)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp

9.103 QERecordSpec Class Reference

Public Member Functions

- **QERecordSpec** (const QString recordType)
- QString **getRecordType** ()
- QString **getFieldName** (const int index)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp

9.104 QERecordSpecList Class Reference

Public Member Functions

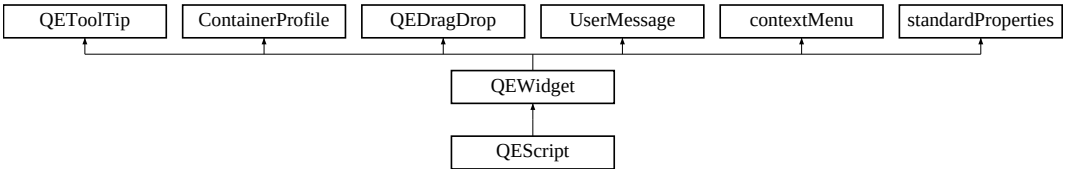
- QERecordSpec ∗ **find** (const QString recordType)
- void **appendOrReplace** (QERecordSpec ∗recordSpec)
- bool **processRecordSpecFile** (const QString &filename)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp

9.105 QEScript Class Reference

Inheritance diagram for QEScript:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEScript** (QWidget *pParent=0)
- void **setDirectoryPath** (QString pValue)
- QString **getDirectoryPath** ()
- void **setShowDirectoryPath** (bool pValue)
- bool **getShowDirectoryPath** ()
- void **setShowDirectoryBrowser** (bool pValue)
- bool **getShowDirectoryBrowser** ()
- void **setShowRefresh** (bool pValue)
- bool **getShowRefresh** ()
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnSize** (bool pValue)
- bool **getShowColumnSize** ()
- void **setShowColumnFilename** (bool pValue)
- bool **getShowColumnFilename** ()
- void **setShowFileExtension** (bool pValue)
- bool **getShowFileExtension** ()
- void **setFileFilter** (QString pValue)
- QString **getFileFilter** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **updateTable** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()

Protected Attributes

- QLineEdit * **qlineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetScript * **qTableWidgetScript**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

Properties

- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**
- detailsLayoutProperty **detailsLayout**

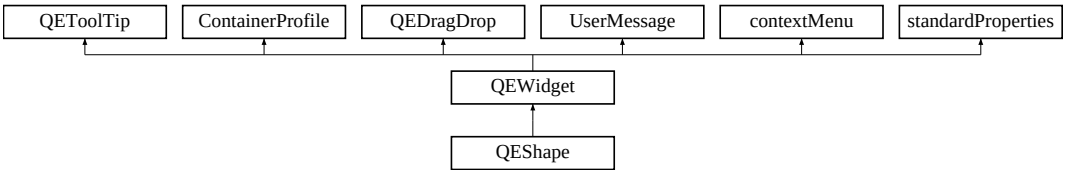
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.h
- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.106 QEShape Class Reference

`#include <QEShape.h>`

Inheritance diagram for QEShape:



Public Types

- enum shapeOptions {

  Line, **Points**, **Polyline**, **Polygon**,

  Rect, **RoundedRect**, **Ellipse**, **Arc**,

  Chord, **Pie**, **Path** }
- enum animationOptions {

  Width, **Height**, **X**, **Y**,

  Transperency, **Rotation**, **ColourHue**, **ColourSaturation**,

  ColourValue, **ColourIndex**, **Penwidth** }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void dbValueChanged1 (const qlonglong &out)
- void dbValueChanged2 (const qlonglong &out)
- void dbValueChanged3 (const qlonglong &out)
- void dbValueChanged4 (const qlonglong &out)
- void dbValueChanged5 (const qlonglong &out)
- void dbValueChanged6 (const qlonglong &out)

Public Member Functions

- QEShape (QWidget *parent=0)
- QEShape (const QString &variableName, QWidget *parent=0)
- void scaleBy (const int m, const int d)

  Scale the widgets my m/d.
- void setAnimation (animationOptions animation, const int index)

  Access function for #animation' properties - refer to animation' properties for details.
- animationOptions getAnimation (const int index)

  Access function for #animation' properties - refer to animation' properties for details.
- void setScale (const double scale, const int index)

  Access function for #scale' properties - refer to scale' properties for details.
- double getScale (const int index)

  Access function for #scale' properties - refer to scale' properties for details.
- void setOffset (const double offset, const int index)

  Access function for #offset' properties - refer to offset' properties for details.
- double getOffset (const int index)

  Access function for #offset' properties - refer to offset' properties for details.
- void setBorder (const bool border)

  Access function for #border' properties - refer to border' properties for details.
- bool getBorder ()

  Access function for #border' properties - refer to border' properties for details.
- void setFill (const bool fill)

  Access function for #fill' properties - refer to fill' properties for details.
- bool getFill ()

  Access function for #fill' properties - refer to fill' properties for details.
- void setShape (shapeOptions shape)

  Access function for #shape' properties - refer to shape' properties for details.
- shapeOptions getShape ()

  Access function for #shape' properties - refer to shape' properties for details.
- void setNumPoints (const unsigned int numPoints)

  Access function for #number of points' properties - refer to number of points' properties for details.
- unsigned int getNumPoints ()

Access function for #number of points' properties - refer to number of points' properties for details.

- void setOriginTranslation (const QPoint originTranslation)

  Access function for #origin translation' properties - refer to origin translation' properties for details.
- QPoint getOriginTranslation ()

  Access function for #origin translation' properties - refer to origin translation' properties for details.
- void setPoint (const QPoint point, const int index)

  Access function for #point' properties - refer to point' properties for details.
- QPoint getPoint (const int index)

  Access function for #point' properties - refer to point' properties for details.
- void setColor (const QColor color, const int index)

  Access function for #colour' properties - refer to colour' properties for details.
- QColor getColor (const int index)

  Access function for #colour' properties - refer to colour' properties for details.
- void setDrawBorder (const bool drawBorder)

  Access function for #draw border' properties - refer to draw border' properties for details.
- bool getDrawBorder ()

  Access function for #draw border' properties - refer to draw border' properties for details.
- void setLineWidth (const unsigned int lineWidth)

  Access function for #line width' properties - refer to line width' properties for details.
- unsigned int getLineWidth ()

  Access function for #line width' properties - refer to line width' properties for details.
- void setStartAngle (const double startAngle)

  Access function for #start angle' properties - refer to start angle' properties for details.
- double getStartAngle ()

  Access function for #start angle' properties - refer to start angle' properties for details.
- void setRotation (const double rotation)

  Access function for #rotation' properties - refer to rotation' properties for details.
- double getRotation ()

  Access function for #rotation' properties - refer to rotation' properties for details.
- void setArcLength (const double arcLength)

  Access function for #arc length' properties - refer to arc length' properties for details.
- double getArcLength ()

  Access function for #arc length' properties - refer to arc length' properties for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

Properties

- QString variable1
- QString variable2
- QString variable3
- QString variable4
- QString variable5
- QString variable6
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- animationOptions animation1
- animationOptions animation2
- animationOptions animation3
- animationOptions animation4
- animationOptions animation5
- animationOptions animation6
- double scale1

  Scale factor applied to data from the 1st variable before it is used to animate the shape.
- double scale2
- double scale3
- double scale4
- double scale5
- double scale6
- double offset1
- double offset2
- double offset3
- double offset4
- double offset5
- double offset6
- QPoint point1
- QPoint point2

- QPoint point3
- QPoint point4
- QPoint point5
- QPoint point6
- QPoint point7
- QPoint point8
- QPoint point9
- QPoint point10
- QColor color1
- QColor color2
- QColor color3
- QColor color4
- QColor color5
- QColor color6
- QColor color7
- QColor color8
- QColor color9
- QColor color10

9.106.1 Detailed Description

This class is a EPICS aware shape widget based on the Qt widget. One of several shapes can be drawn within the widget, and up to 6 variables can be used to animate various attributes of the shape. For example to represent beam positino and size, an elipse can be drawn with four variables animating its vertcal and horizontal size and position. It is tighly integrated with the base class QEWidget which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.106.2 Member Enumeration Documentation

9.106.2.1 enum QEShape::animationOptions

Options for how a variable will animate the shape.

9.106.2.2 enum QEShape::shapeOptions

Options for the type of shape.

9.106.2.3 enum QEShape::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.106.3 Constructor & Destructor Documentation

9.106.3.1 QEShape::QEShape (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutionsProperty() to define a variable and, optionally, macro substitutions later.

9.106.3.2 QEShape::QEShape (const QString & *variableName,* QWidget * *parent* = 0)

Create with a single variable. (Note, the QEShape widget can use up to 6 variables) A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.106.4 Member Function Documentation

9.106.4.1 void QEShape::dbValueChanged1 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the first variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.4.2 void QEShape::dbValueChanged2 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the second variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.4.3 void QEShape::dbValueChanged3 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the third variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.4.4 void QEShape::dbValueChanged4 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the fourth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.4.5 void QEShape::dbValueChanged5 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the fifth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.4.6 void QEShape::dbValueChanged6 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the sixth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.106.5 Property Documentation

9.106.5.1 bool QEShape::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.106.5.2 animationOptions QEShape::animation1 [read, write]

Animation to be effected by the 1st variable. This is used to select what the effect changing data for the 1st variable will have on the shape.

9.106.5.3 animationOptions QEShape::animation2 [read, write]

Animation to be effected by the 2nd variable. This is used to select what the effect changing data for the 2nd variable will have on the shape.

9.106.5.4 animationOptions QEShape::animation3 [read, write]

Animation to be effected by the 3rd variable. This is used to select what the effect changing data for the 3rd variable will have on the shape.

9.106.5.5 animationOptions QEShape::animation4 [read, write]

Animation to be effected by the 4th variable. This is used to select what the effect changing data for the 4th variable will have on the shape.

9.106.5.6 animationOptions QEShape::animation5 [read, write]

Animation to be effected by the 5th variable. This is used to select what the effect changing data for the 5th variable will have on the shape.

9.106.5.7 animationOptions QEShape::animation6 [read, write]

Animation to be effected by the 6th variable. This is used to select what the effect changing data for the 6th variable will have on the shape.

9.106.5.8 QColor QEShape::color1 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.9 QColor QEShape::color10 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.10 QColor QEShape::color2 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.11 QColor QEShape::color3 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.12 QColor QEShape::color4 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.13 QColor QEShape::color5 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.14 QColor QEShape::color6 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.15 QColor QEShape::color7 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.16 QColor QEShape::color8 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.17 QColor QEShape::color9 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.106.5.18 bool QEShape::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.106.5.19 unsigned QEShape::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

The number of points to use when drawing shapes that are defined by a variable number of points, such as polyline, polygon, path, and series of points.

Sets the width of the pen. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path

9.106.5.20 double QEShape::offset1 [read, write]

Offset applied to data from the 1st variable before it is used to animate the shape

9.106.5.21 double QEShape::offset2 [read, write]

Offset applied to data from the 2nd variable before it is used to animate the shape

9.106.5.22 double QEShape::offset3 [read, write]

Offset applied to data from the 3rd variable before it is used to animate the shape

9.106.5.23 double QEShape::offset4 [read, write]

Offset applied to data from the 4th variable before it is used to animate the shape

9.106.5.24 double QEShape::offset5 [read, write]

Offset applied to data from the 5th variable before it is used to animate the shape

9.106.5.25 double QEShape::offset6 [read, write]

Offset applied to data from the 6th variable before it is used to animate the shape

9.106.5.26 QPoint QEShape::point1 [read, write]

1st coordinate used when drawing the shape. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path, Text, Pixmap

9.106.5.27 QPoint QEShape::point10 [read, write]

10th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.28 QPoint QEShape::point2 [read, write]

2nd coordinate used when drawing the shape. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path, Pixmap

9.106.5.29 QPoint QEShape::point3 [read, write]

3rd coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.30 QPoint QEShape::point4 [read, write]

4th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.31 QPoint QEShape::point5 [read, write]

5th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.32 QPoint QEShape::point6 [read, write]

6th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.33 QPoint QEShape::point7 [read, write]

7th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.34 QPoint QEShape::point8 [read, write]

8th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.35 QPoint QEShape::point9 [read, write]

9th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.106.5.36 double QEShape::scale2 [read, write]

Scale factor applied to data from the 2nd variable before it is used to animate the shape

9.106.5.37 double QEShape::scale3 [read, write]

Scale factor applied to data from the 3rd variable before it is used to animate the shape

9.106.5.38 double QEShape::scale4 [read, write]

Scale factor applied to data from the 4th variable before it is used to animate the shape

9.106.5.39 double QEShape::scale5 [read, write]

Scale factor applied to data from the 5th variable before it is used to animate the shape

9.106.5.40 double QEShape::scale6 [read, write]

Scale factor applied to data from the 6th variable before it is used to animate the shape

9.106.5.41 UserLevels QEShape::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.106.5.42 QString QEShape::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.106.5.43 QString QEShape::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.106.5.44 QString QEShape::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.106.5.45 UserLevels QEShape::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.106.5.46 QString QEShape::variable1 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale1 and offset1 then the attribute selected for animation is selected by the property animation1.

9.106.5.47 QString QEShape::variable2 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale2 and offset2 then the attribute selected for animation is selected by the property animation2.

9.106.5.48 QString QEShape::variable3 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale3 and offset3 then the attribute selected for animation is selected by the property animation3.

9.106.5.49 QString QEShape::variable4 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale4 and offset4 then the attribute selected for animation is selected by the property animation4.

9.106.5.50 QString QEShape::variable5 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale5 and offset5 then the attribute selected for animation is selected by the property animation5.

9.106.5.51 QString QEShape::variable6 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale6 and offset6 then the attribute selected for animation is selected by the property animation6.

9.106.5.52 bool QEShape::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.106.5.53 QString QEShape::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.106.5.54 bool QEShape::visible [read, write]

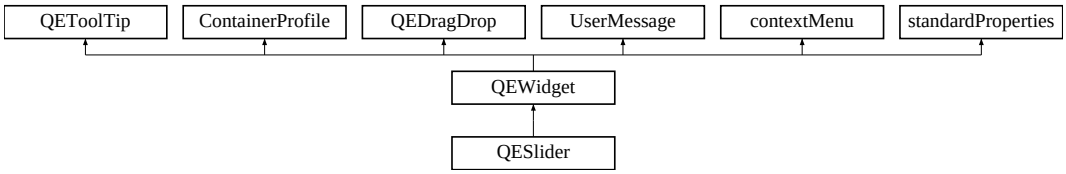
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEShape/QEShape.h
- /tmp/epicsqt/trunk/framework/widgets/QEShape/QEShape.cpp

9.107 QESlider Class Reference

Inheritance diagram for QESlider:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void dbValueChanged (const qlonglong &out)

Public Member Functions

- **QESlider** (QWidget *parent=0)
- **QESlider** (const QString &variableName, QWidget *parent=0)
- void **setWriteOnChange** (bool writeOnChange)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)

- bool **getSubscribe** ()
- void **setScale** (double scaleIn)
- double **getScale** ()
- void **setOffset** (double offsetIn)
- double **getOffset** ()
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool writeOnChange

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.107.1 Member Enumeration Documentation

9.107.1.1 enum QESlider::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.107.2 Member Function Documentation

9.107.2.1 void QESlider::dbValueChanged (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.107.3 Member Data Documentation

9.107.3.1 bool QESlider::writeOnChange [read, write, protected]

Sets if this widget writes any changes as the user moves the slider (the QSlider 'valueChanged' signal is emitted). Default is 'true' (writes any changes when the QSlider 'valueChanged' signal is emitted).

9.107.4 Property Documentation

9.107.4.1 bool QESlider::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.107.4.2 bool QESlider::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.107.4.3 unsigned QESlider::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.107.4.4 bool QESlider::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.107.4.5 UserLevels QESlider::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.107.4.6 QString QESlider::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.107.4.7 QString QESlider::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.107.4.8 QString QESlider::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example,

'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.107.4.9 UserLevels QESlider::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.107.4.10 QString QESlider::variable [read, write]

EPICS variable name (CA PV)

9.107.4.11 bool QESlider::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.107.4.12 QString QESlider::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.107.4.13 bool QESlider::visible [read, write]

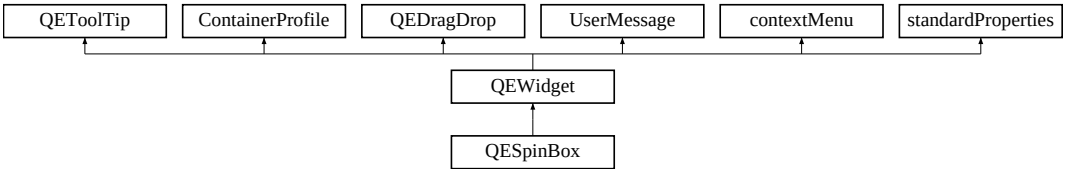
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QESlider/QESlider.h
- /tmp/epicsqt/trunk/framework/widgets/QESlider/QESlider.cpp

9.108 QESpinBox Class Reference

Inheritance diagram for QESpinBox:



Public Types

- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void dbValueChanged (const double &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.

Public Member Functions

- **QESpinBox** (QWidget *parent=0)
- **QESpinBox** (const QString &variableName, QWidget *parent=0)
- void **setWriteOnChange** (bool writeOnChangeIn)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setAddUnitsAsSuffix** (bool addUnitsAsSuffixIn)
- bool **getAddUnitsAsSuffix** ()
- void **setUseDbPrecisionForDecimals** (bool useDbPrecisionForDecimalIn)
- bool **getUseDbPrecisionForDecimals** ()
- UserLevels getUserLevelVisibilityProperty ()

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for userLevelVisibility property - refer to userLevelVisibility property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for userLevelEnabled property - refer to userLevelEnabled property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **writeOnChange**
- bool **addUnitsAsSuffix**
- bool **useDbPrecisionForDecimal**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState
- bool subscribe
- bool **useDbPrecision**
- bool **addUnits**

9.108.1 Member Enumeration Documentation

9.108.1.1 enum QESpinBox::UserLevels

User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.108.2 Member Function Documentation

9.108.2.1 void QESpinBox::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.108.3 Property Documentation

9.108.3.1 bool QESpinBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.108.3.2 bool QESpinBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from standardProperties.

9.108.3.3 unsigned QESpinBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.108.3.4 bool QESpinBox::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.108.3.5 UserLevels QESpinBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.108.3.6 QString QESpinBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.108.3.7 QString QESpinBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.108.3.8 QString QESpinBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.108.3.9 UserLevels QESpinBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.108.3.10 QString QESpinBox::variable [read, write]

EPICS variable name (CA PV)

9.108.3.11 bool QESpinBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEToolTip.

9.108.3.12 QString QESpinBox::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.108.3.13 bool QESpinBox::visible [read, write]

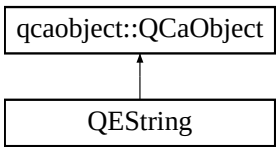
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QESpinBox/QESpinBox.h
- /tmp/epicsqt/trunk/framework/widgets/QESpinBox/QESpinBox.cpp

9.109 QEString Class Reference

Inheritance diagram for QEString:



Public Slots

- void **writeString** (const QString &data)

Signals

- void **stringConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **stringChanged** (const QString &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn)
- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)
- bool **writeString** (const QString &data, QString &message)

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

- /tmp/epicsqt/trunk/framework/data/include/QEString.h
- /tmp/epicsqt/trunk/framework/data/src/QEString.cpp

9.110 QEStringFormatting Class Reference

Public Types

- enum formats {
  FORMAT_DEFAULT, FORMAT_FLOATING, FORMAT_INTEGER, FORMAT_UNSIGNEDINTEGER, FORMAT_TIME, FORMAT_LOCAL_ENUMERATE, FORMAT_STRING }
- enum notations { NOTATION_FIXED = QTextStream::FixedNotation, NOTATION_SCIENTIFIC = QTextStream::ScientificNotation, NOTATION_AUTOMATIC = QTextStream::SmartNotation }
- enum arrayActions { APPEND, ASCII, INDEX }

Public Member Functions

- QString **formatString** (const QVariant &value)
- QVariant **formatValue** (const QString &text, bool &ok)
- void **setDbEgu** (QString egu)
- void **setDbEnumerations** (QStringList enumerations)
- void **setDbPrecision** (unsigned int dbPrecisionIn)
- void **setPrecision** (int precision)
- void **setUseDbPrecision** (bool useDbPrecision)
- void **setLeadingZero** (bool leadingZero)
- void **setTrailingZeros** (bool trailingZeros)
- void **setFormat** (formats format)

- void **setRadix** (unsigned int radix)
- void **setNotation** (notations notation)
- void **setArrayAction** (arrayActions arrayActionIn)
- void **setArrayIndex** (unsigned int arrayIndexIn)
- void **setAddUnits** (bool addUnits)
- void **setLocalEnumeration** (QString localEnumerationIn)
- int **getPrecision** ()
- bool **getUseDbPrecision** ()
- bool **getLeadingZero** ()
- bool **getTrailingZeros** ()
- formats **getFormat** ()
- unsigned int **getRadix** ()
- notations **getNotation** ()
- arrayActions **getArrayAction** ()
- unsigned int **getArrayIndex** ()
- bool **getAddUnits** ()
- QString **getLocalEnumeration** ()

9.110.1 Member Enumeration Documentation

9.110.1.1 enum QEStringFormatting::arrayActions

What action to take when formatting array data

Enumerator:

APPEND Interpret each element in the array as an unsigned integer and append string representations of each element from the array with a space in between each.

ASCII Interpret each element from the array as a character in a string. Translate all non printing characters to '?' except for trailing zeros (ignore them)

INDEX Interpret the element selected by setArrayIndex() as an unsigned integer.

9.110.1.2 enum QEStringFormatting::formats

Formatting options

Enumerator:

FORMAT_DEFAULT Format according to the EPICS database record type.

FORMAT_FLOATING Format as a floating point number.

FORMAT_INTEGER Format as an integer.

FORMAT_UNSIGNEDINTEGER Format as an unsigned integer.

FORMAT_TIME Format as a time.

FORMAT_LOCAL_ENUMERATE Format as a selection from the local enumerations set by setLocalEnumeration()

FORMAT_STRING Format as a string.

9.110.1.3 enum QEStringFormatting::notations

Notations when formatting a floating point number

Enumerator:

NOTATION_FIXED Standard floating point 123456.789.

NOTATION_SCIENTIFIC Scientific representation 1.23456789e6.

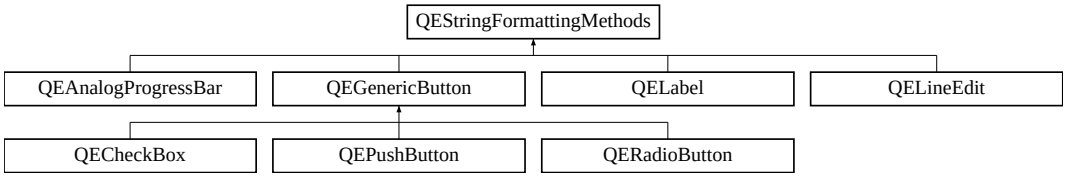
NOTATION_AUTOMATIC Automatic choice of standard or scientific notation.

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

- /tmp/epicsqt/trunk/framework/data/include/QEStringFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEStringFormatting.cpp

9.111 QEStringFormattingMethods Class Reference

Inheritance diagram for QEStringFormattingMethods:



Public Member Functions

- virtual void **stringFormattingChange** ()=0
- void **setPrecision** (int precision)
- int **getPrecision** ()
- void **setUseDbPrecision** (bool useDbPrecision)
- bool **getUseDbPrecision** ()
- void **setLeadingZero** (bool leadingZero)
- bool **getLeadingZero** ()
- void **setTrailingZeros** (bool trailingZeros)
- bool **getTrailingZeros** ()
- void **setAddUnits** (bool addUnits)
- bool **getAddUnits** ()
- void **setLocalEnumeration** (QString localEnumeration)
- QString **getLocalEnumeration** ()
- void **setFormat** (QEStringFormatting::formats format)
- QEStringFormatting::formats **getFormat** ()
- void **setRadix** (unsigned int radix)
- unsigned int **getRadix** ()

- void **setNotation** (QEStringFormatting::notations notation)
- QEStringFormatting::notations **getNotation** ()
- void **setArrayAction** (QEStringFormatting::arrayActions arrayAction)
- QEStringFormatting::arrayActions **getArrayAction** ()
- void **setArrayIndex** (unsigned int arrayIndex)
- unsigned int **getArrayIndex** ()

Protected Attributes

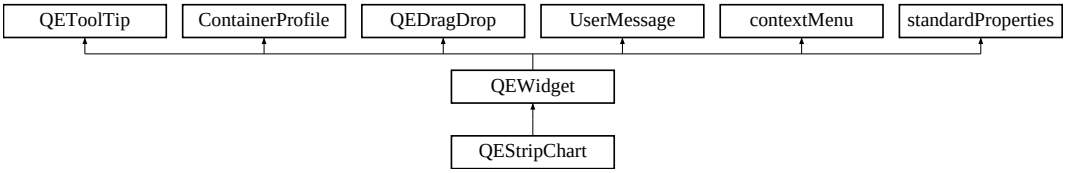
- QEStringFormatting **stringFormatting**

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEStringFormattingMethods.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEStringFormattingMethods.cpp

9.112 QEStripChart Class Reference

Inheritance diagram for QEStripChart:



Classes

- class PrivateData

Public Types

- enum **Constants** { **NUMBER_OF_PVS** = 12 }

Public Member Functions

- **QEStripChart** (QWidget *parent=0)
- QSize **sizeHint** () const
- QDateTime **getStartDateTime** ()
- QDateTime **getEndDateTime** ()
- void **setEndDateTime** (QDateTime endDateTimeIn)
- int **getDuration** ()

- void **setDuration** (int durationIn)
- double **getYMinimum** ()
- void **setYMinimum** (const double yMinimumIn)
- double **getYMaximum** ()
- void **setYMaximum** (const double yMaximumIn)
- void **setYRange** (const double yMinimumIn, const double yMaximumIn)
- void **plotData** ()
- void **addToPredefinedList** (const QString &pvName)
- QString **getPredefinedItem** (int i)

Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- void **setup** ()
- qcaobject::QCaObject ∗ **createQcaItem** (unsigned int variableIndex)
- void **establishConnection** (unsigned int variableIndex)
- void saveConfiguration (PersistanceManager ∗pm)
- void restoreConfiguration (PersistanceManager ∗pm, restorePhases restorePhase)

Properties

- int **duration**
- double **yMinimum**
- double **yMaximum**
- QString **variable1**
- QString **variable2**
- QString **variable3**
- QString **variable4**
- QString **variable5**
- QString **variable6**
- QString **variable7**
- QString **variable8**
- QString **variable9**
- QString **variable10**
- QString **variable11**
- QString **variable12**
- QString variableSubstitutions
- QColor **colour1**
- QColor **colour2**
- QColor **colour3**
- QColor **colour4**
- QColor **colour5**
- QColor **colour6**

- QColor **colour7**
- QColor **colour8**
- QColor **colour9**
- QColor **colour10**
- QColor **colour11**
- QColor **colour12**

Friends

- class **PrivateData**
- class QEStripChartItem

9.112.1 Member Function Documentation

9.112.1.1 void QEStripChart::restoreConfiguration (PersistanceManager * , restorePhases) [protected, virtual]

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the PersistanceManager. For example, a QEStripChart may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented from QEWidget.

9.112.1.2 void QEStripChart::saveConfiguration (PersistanceManager *) [protected, virtual]

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the PersistanceManager. For example, a QEStripChart may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented from QEWidget.

9.112.2 Property Documentation

9.112.2.1 QString QEStripChart::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.113 QEStripChartAdjustPVDialog Class Reference

Public Member Functions

- **QEStripChartAdjustPVDialog** (QWidget *parent=0)
- void **setValueScaling** (const ValueScaling &valueScale)
- ValueScaling **getValueScaling** ()
- void **setSupport** (const double min, const double max, const TrackRange &loprHopr, const TrackRange &plotted, const TrackRange &buffered)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialog.cpp

9.114 QEStripChartContextMenu Class Reference

Signals

- void **contextMenuSelected** (const unsigned int, const QEStripChartNames::ContextMenuOptions)

Public Member Functions

- QEStripChartContextMenu (bool inUse, QWidget *parent=0)
- void **setPredefinedNames** (const QStringList &pvList)
- void **setUseReceiveTime** (const bool useReceiveTime)
- void **setArchiveReadHow** (const QEArchiveInterface::How how)
- void **setLineDrawMode** (const QEStripChartNames::LineDrawModes mode)
- QAction * **exec** (const unsigned int slot, const QPoint &pos, QAction *at=0)

Static Public Attributes

- static const int **numberPrefefinedItems** = 10

9.114.1 Constructor & Destructor Documentation

9.114.1.1 QEStripChartContextMenu::QEStripChartContextMenu (bool *inUse,* QWidget * *parent* = 0) [explicit]

Construct strip chart item context menu. This menu item creates all required sub menu items. inUse set true for an inuse slot, i.e. already has a PV allocated. inUse set false for an empty slot.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartContextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartContextMenu.cpp

9.115 QEStripChartItem Class Reference

Classes

- class PrivateData

Public Slots

- void **setColour** (const QColor &colour)

Signals

- void **itemContextMenuRequested** (const unsigned int, const QPoint &)

Public Member Functions

- **QEStripChartItem** (QEStripChart ∗chart, QLabel ∗pvName, QELabel ∗caLabel, unsigned int slot)
- bool **isInUse** ()
- void **setPvName** (QString pvName, QString substitutions)
- QString **getPvName** ()
- void **setScaling** (const double d, const double m, const double c)
- void **getScaling** (double &d, double &m, double &c)
- bool **isScaled** ()
- bool **getUseReceiveTime** ()
- QEArchiveInterface::How **getArchiveReadHow** ()
- QEStripChartNames::LineDrawModes **getLineDrawMode** ()
- QColor **getColour** ()
- TrackRange **getLoprHopr** (bool doScale)
- TrackRange **getDisplayedMinMax** (bool doScale)
- TrackRange **getBufferedMinMax** (bool doScale)
- void **readArchive** ()
- void **normalise** ()
- void **plotData** (const double timeScale, const QEStripChartNames::YScaleModes yScaleMode)
- void **contextMenuSelected** (const QEStripChartNames::ContextMenuOptions option)
- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

Public Attributes

- QCaVariableNamePropertyManager **pvNameProperyManager**

Protected Member Functions

- bool **eventFilter** (QObject ∗obj, QEvent ∗event)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItem.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.116 QEStripChartItemDialog Class Reference

Public Member Functions

- **QEStripChartItemDialog** (QWidget ∗parent=0)
- void **setPvName** (QString pvNameIn)
- QString **getPvName** ()
- bool **isClear** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItemDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItemDialog.cpp

9.117 QEStripChartNames Class Reference

Public Types

- enum **ChartTimeModes** { **tmRealTime**, **tmPaused**, **tmHistorical** }
- enum **ChartYRanges** {
  manual, **operatingRange**, **plotted**, **buffered**,
  dynamic, **normalised** }
- enum **PlayModes** {
  play, **pause**, **forward**, **backward**,
  selectTimes }
- enum **StateModes** { **previous**, **next** }
- enum **VideoModes** { **normal**, **reverse** }
- enum **YScaleModes** { **linear**, **log** }
- enum **LineDrawModes** { **ldmHide**, **ldmRegular**, **ldmBold** }

- enum **ContextMenuOptions** {

  SCCM_NONE = contextMenu::CM_SPECIFIC_WIDGETS_START_HERE, **SCCM_-COPY_PV_NAMES**, **SCCM_PASTE_PV_NAMES**, **SCCM_READ_ARCHIVE**,

  SCCM_SCALE_CHART_AUTO, **SCCM_SCALE_CHART_PLOTTED**, **SCCM_-SCALE_CHART_BUFFERED**, **SCCM_SCALE_PV_RESET**,

  SCCM_SCALE_PV_GENERAL, **SCCM_SCALE_PV_AUTO**, **SCCM_SCALE_-PV_PLOTTED**, **SCCM_SCALE_PV_BUFFERED**,

  SCCM_SCALE_PV_CENTRE, **SCCM_PLOT_RECTANGULAR**, **SCCM_PLOT_-SMOOTH**, **SCCM_PLOT_SERVER_TIME**,

  SCCM_PLOT_CLIENT_TIME, **SCCM_ARCH_LINEAR**, **SCCM_ARCH_PLOTBIN**, **SCCM_ARCH_RAW**,

  SCCM_ARCH_SHEET, **SCCM_ARCH_AVERAGED**, **SCCM_LINE_HIDE**, **SCCM_-LINE_REGULAR**,

  SCCM_LINE_BOLD, **SCCM_LINE_COLOUR**, **SCCM_PV_EDIT_NAME**, **SCCM_-ADD_TO_PREDEFINED**,

  SCCM_PV_WRITE_TRACE, **SCCM_PV_STATS**, **SCCM_PV_CLEAR**, **SCCM_-PV_ADD_NAME**,

  SCCM_PV_PASTE_NAME, **SCCM_PREDEFINED_01**, **SCCM_PREDEFINED_-02**, **SCCM_PREDEFINED_03**,

  SCCM_PREDEFINED_04, **SCCM_PREDEFINED_05**, **SCCM_PREDEFINED_-06**, **SCCM_PREDEFINED_07**,

  SCCM_PREDEFINED_08, **SCCM_PREDEFINED_09**, **SCCM_PREDEFINED_-10** }

Static Public Attributes

- static const ContextMenuOptions **ContextMenuItemFirst** = SCCM_READ_ARCHIVE
- static const ContextMenuOptions **ContextMenuItemLast** = SCCM_PREDEFINED_-10

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartNames.h

9.118 QEStripChartRangeDialog Class Reference

Public Member Functions

- **QEStripChartRangeDialog** (QWidget ∗parent=0)
- void **setRange** (const double min, const double max)
- double **getMinimum** ()
- double **getMaximum** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartRangeDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartRangeDialog.cpp

9.119 QEStripChartTimeDialog Class Reference

Public Member Functions

- **QEStripChartTimeDialog** (QWidget ∗parent=0)
- void **setMaximumDateTime** (QDateTime datetime)
- void **setStartDateTime** (QDateTime datetime)
- QDateTime **getStartDateTime** ()
- void **setEndDateTime** (QDateTime datetime)
- QDateTime **getEndDateTime** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.cpp

9.120 QEStripChartToolBar Class Reference

This class holds all the StripChart tool bar widgets.

`#include <QEStripChartToolBar.h>`

Classes

- class OwnWidgets

Signals

- void **stateSelected** (const QEStripChartNames::StateModes mode)
- void **videoModeSelected** (const QEStripChartNames::VideoModes mode)
- void **yScaleModeSelected** (const QEStripChartNames::YScaleModes mode)
- void **yRangeSelected** (const QEStripChartNames::ChartYRanges scale)
- void **durationSelected** (const int seconds)
- void **timeZoneSelected** (const Qt::TimeSpec timeSpec)
- void **playModeSelected** (const QEStripChartNames::PlayModes mode)
- void **readArchiveSelected** ()

Public Member Functions

- **QEStripChartToolBar** (QWidget ∗parent=0)
- void **setTimeStatus** (const QString &timeStatus)
- void **setStateSelectionEnabled** (const QEStripChartNames::StateModes mode, const bool enabled)

Static Public Attributes

- static const int **designHeight** = 44

Protected Member Functions

- void **resizeEvent** (QResizeEvent ∗event)

9.120.1 Detailed Description

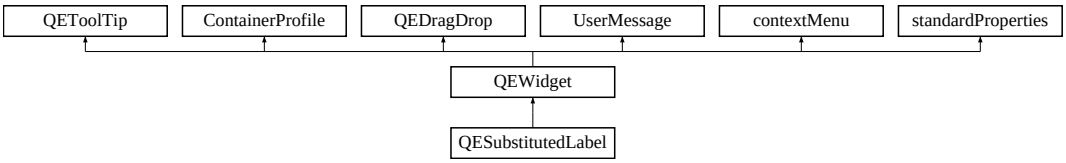
This class holds all the StripChart tool bar widgets.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.121 QESubstitutedLabel Class Reference

Inheritance diagram for QESubstitutedLabel:



Public Member Functions

- **QESubstitutedLabel** (QWidget ∗parent=0)
- void **establishConnection** (unsigned int variableIndex)
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- QString **getLabelTextPropertyFormat** ()
- void **setLabelTextPropertyFormat** (QString labelTextIn)

Protected Attributes

- QString labelText

Properties

- QString textSubstitutions

9.121.1 Member Data Documentation

9.121.1.1 QString QESubstitutedLabel::labelText [read, write, protected]

Label text to be substituted. This text will be copied to the label text after applying any macro substitutions from the textSubstitutions property

9.121.2 Property Documentation

9.121.2.1 QString QESubstitutedLabel::textSubstitutions [read, write]

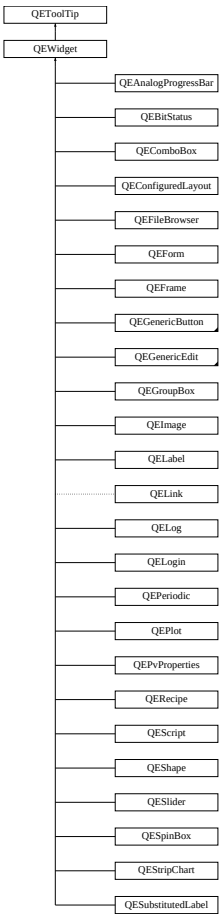
Text substitutions. These substitutions are applied to the 'labelText' property prior to copying it to the label text.

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

- /tmp/epicsqt/trunk/framework/widgets/QESubstitutedLabel/QESubstitutedLabel.h
- /tmp/epicsqt/trunk/framework/widgets/QESubstitutedLabel/QESubstitutedLabel.cpp

9.122 QEToolTip Class Reference

Inheritance diagram for QEToolTip:



Public Member Functions

- **QEToolTip** (QWidget ∗ownerIn)
- void **updateToolTipVariable** (const QString &variable)
- void **updateToolTipAlarm** (const QString &alarm)
- void **updateToolTipCustom** (const QString &custom)
- void **updateToolTipConnection** (bool connection)
- void **setVariableAsToolTip** (bool variableAsToolTip)
- bool **getVariableAsToolTip** ()

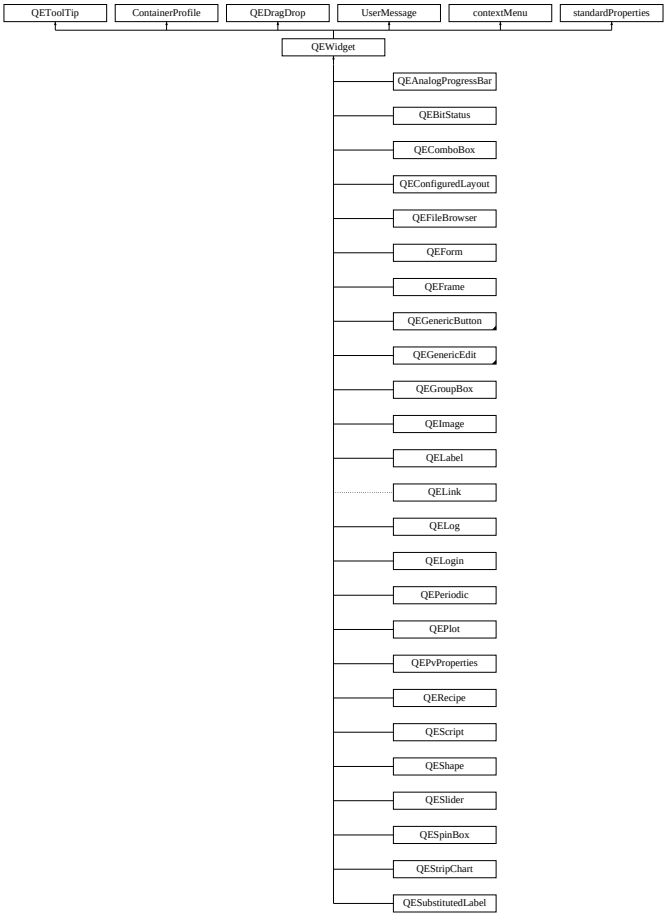
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/widgets/include/QEToolTip.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEToolTip.cpp

9.123 QEWidget Class Reference

`#include <QEWidget.h>`

Inheritance diagram for QEWidget:



Public Types

- enum restorePhases { **APPLICATION** = SaveRestoreSignal::RESTORE_APPLICATION, **FRAMEWORK** = SaveRestoreSignal::RESTORE_QEFRAMEWORK }

  Restore phases. When a widget's persistant data is restored, the restore occurs in two phases.

Public Member Functions

- QEWidget (QWidget ∗ownerIn)

  Constructor.

- virtual ~QEWidget ()

  Destructor.
- void activate ()
- void deactivate ()
- unsigned int getMessageSourceId ()
- void setMessageSourceId (unsigned int messageSourceId)
- qcaobject::QCaObject * getQcaItem (unsigned int variableIndex)
- void setupContextMenu (QWidget *w)
- QColor getColor (QCaAlarmInfo &alarmInfo, const int saturation)
- void processAlarmInfo (QCaAlarmInfo &alarmInfo)
- void readNow ()
- virtual void writeNow ()
- virtual void setVariableNameAndSubstitutions (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- QFile * openQEFile (QString name, QFile::OpenModeFlag mode)
- QString defaultFileLocation ()
- QString getFrameworkVersion ()
- virtual void saveConfiguration (PersistanceManager *)
- virtual void restoreConfiguration (PersistanceManager *, restorePhases)
- virtual void scaleBy (const int, const int)

Static Public Member Functions

- static QFile * findQEFile (QString name, ContainerProfile *profile)
- static bool **inDesigner** ()

Protected Member Functions

- void **setNumVariables** (unsigned int numVariablesIn)
- qcaobject::QCaObject * **createConnection** (unsigned int variableIndex)
- virtual qcaobject::QCaObject * **createQcaItem** (unsigned int variableIndex)
- virtual void **establishConnection** (unsigned int variableIndex)
- QString **persistantName** (QString prefix)

Protected Attributes

- bool **subscribe**

9.123.1 Detailed Description

This class is used as a base for all CA aware wigets, such as QELabel, QESpinBox, etc. It manages common issues including creating a source of CA data updates, handling error, warning and status messages, and setting tool tips based on variable names.

Note, there is tight integration between the CA aware widget classes, this class, and its base classes, especially VariableNameManager and QEToolTip.

In particular, this class manages QCaObject classes that stream updates to the CA aware widget class. But this class, however, doesn't know how to format the data, or how the updates will be used. To resolve this, this class asks its parent class (such as QELabel) to create the QCaObject class in what ever flavour it wants, by calling the virtual function createQcaItem. A QELabel, for example, wants string updates so it creates a QEString which is based on a QCaObject class and formats all updates as strings.

The CA aware parent class (such as QELabel) defines a variable by calling VariableNameManager::setVariableName(). The VariableNamePropertyManager class calls the establishConnection function of the CA aware parent class, such as QELabel when it has a new variable name.

This class uses its base QEToolTip class to format tool tips. that class in turn calls the CA aware parent class (such as QELabel) directly to make use of a new tool tip.

After construction, a CA aware widget is activated (starts updating) by calling it's establishConnection() function in one of two ways:

1) The variable name or variable name substitutions is changed by calling setVariableName or setVariableNameSubstitutions respectively. These functions are in the VariableNameManager class. The VariableNamePropertyManager calls a virtual function establishConnection() which is implemented by the CA aware widget. This is how a CA aware widget is activated in 'designer'. It occurs when 'designer' updates the variable name property or variable name substitution property.

2) When an QEForm widget is created, resulting in a set of CA aware widgets being created by loading a UI file contining plugin definitions. After loading the plugin widgets, code in the QEForm class calls the activate() function in this class (QEWiget). the activate() function calls establishConnection() in the CA aware widget for each variable. This simulates what the VariableNamePropertyManager does as each variable name is entered (see 1, above, for details)

No matter which way a CA aware widget is activated, the establishConnection() function in the CA aware widget is called for each variable. The establishConnection() function asks this QEWidget base class, by calling the createConnection() function, to perform the tasks common to all CA aware widgets for establishing a stream of CA data.

The createConnection() function sets up the widget 'tool tip', then immedietly calls the CA aware widget back asking it to create an object based on QCaObject. This object will supply a stream of CA update signals to the CA aware object in a form that it needs. For example a QELabel creates a QEString object. The QEString class is based on the QCaObject class and converts all update data to a strings which is required for updating a Qt label widget. This class stores the QCaObject based class.

After the establishConnection() function in the CA aware widget has called createConnection(), the remaining task of the establishConnection() function is to connect the signals of the newly created QCaObject based classes to its own slots so that data updates can be used. For example, a QELabel connects the 'stringChanged' signal from the QEString object to its setLabelText slot.

9.123.2 Member Function Documentation

9.123.2.1 void QEWidget::activate ( )

Initiate updates. Called after all configuration is complete.

9.123.2.2 void QEWidget::deactivate ( )

Terminates updates. This has been provided for third party (non QEGui) applications using the framework.

9.123.2.3 QString QEWidget::defaultFileLocation ( )

Returns the default location to create files. Use this to create files in a consistant location

9.123.2.4 QFile * QEWidget::findQEFile (QString *name,* ContainerProfile * *profile*) [static]

Static method that looks for a file in a standard set of locations Returns a pointer to a QFile which is the caller's responsibility to delete, or NULL if the file was not found.

9.123.2.5 QColor QEWidget::getColor (QCaAlarmInfo & *alarmInfo,* const int *saturation*)

Return a colour to update the widget's look to reflect the current alarm state Note, the color is determined by the alarmInfo class, but since that class is used in non gui applications, it can't return a QColor

9.123.2.6 QString QEWidget::getFrameworkVersion ( )

Returns the QE framework that built this instance of the widget. On windows, the QE-Framework DLL may be loaded twice with potentially different versions of it.

9.123.2.7 unsigned int QEWidget::getMessageSourceId ( ) [inline]

Get the message source ID. The message source ID is used as part of the system where QE widgets can emit a message and have the right QE widget in the right form catch the message. Refer to the UserMessage class for further details.

9.123.2.8 qcaobject::QCaObject * QEWidget::getQcaItem (unsigned int *variableIndex*)

Return a reference to one of the qCaObjects used to stream CA updates

9.123.2.9 QFile * QEWidget::openQEFile (QString *name,* QFile::OpenModeFlag *mode*)

Looks for a file in a standard set of locations (and opens the file)

9.123.2.10 void QEWidget::processAlarmInfo (QCaAlarmInfo & *alarmInfo*)

This convenience function updates the alarm tool tip, and alarm status style if the displayAlarmState property is set to true - assumes the widget uses standard properties. This function is perhaps most usefull for single-variable widgets.

9.123.2.11 void QEWidget::readNow ( )

Perform a single shot read on all variables (Usefull when not subscribing by default)

9.123.2.12 virtual void QEWidget::restoreConfiguration (PersistanceManager * , restorePhases) [inline, virtual]

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the PersistanceManager. For example, a QEStripChart may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented in QEPvProperties, and QEStripChart.

9.123.2.13 virtual void QEWidget::saveConfiguration (PersistanceManager *) [inline, virtual]

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the PersistanceManager. For example, a QEStripChart may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented in QEPvProperties, and QEStripChart.

9.123.2.14 virtual void QEWidget::scaleBy (const int , const int) [inline, virtual]

Any QEWidget that requires additional scaling, i.e. above and beyond the standard scaling applied to size, minimum size, maximum size and font size, may override this function in order to perform any bespoke scaling need by the widget (for example see QEShape). The scaling is defined using a rational number specifed by two integers (m, d). The first (m) parameter is the multiplier and the second (d) parameter is the divisor. For example, if m = 4 and d = 5, then an 80% scaling should be applied. And if m = 5 and d = 4, and a 125% scaling is required.

Reimplemented in QEPvProperties, and QEShape.

9.123.2.15 void QEWidget::setMessageSourceId (unsigned int *messageSourceId*) [inline]

Set the message source ID. The message source ID is used as part of the system where QE widgets can emit a message and have the right QE widget in the right form catch the message. Refer to the UserMessage class for further details.

9.123.2.16 void QEWidget::setupContextMenu (QWidget * *w*)

Take a menu widgt and add it as the context menu for this widget

9.123.2.17 void QEWidget::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) [virtual]

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

Reimplemented in QEBitStatus, and QEForm.

9.123.2.18 virtual void QEWidget::writeNow ( ) [inline, virtual]

(Control widgets only - such as QELineEdit) Write the value now. Used when writeOnChange, writeOnEnter, etc are all false

Reimplemented in QEGenericEdit.

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEWidget.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEWidget.cpp

9.124 QEWidgets Class Reference

Public Member Functions

- **QEWidgets** (QObject *parent=0)
- virtual QList< QDesignerCustomWidgetInterface * > **customWidgets** () const

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEDesignerPlugin.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEDesignerPlugin.cpp

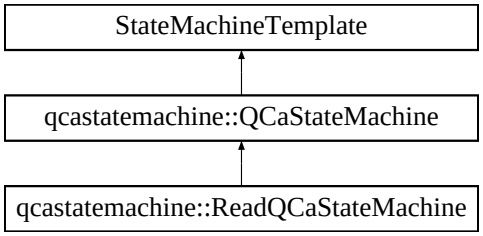
9.125 QLabelList Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.126 qcastatemachine::ReadQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ReadQCaStateMachine:



Public Member Functions

- **ReadQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.127 ROIinfo Class Reference

Public Member Functions

- void **setX** (long x)
- void **setY** (long y)
- void **setW** (long w)
- void **setH** (long h)
- void **clearX** ()
- void **clearY** ()
- void **clearW** ()
- void **clearH** ()
- bool **getStatus** ()
- QRect **getArea** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h

9.128 SaveRestoreSignal Class Reference

Public Types

- enum **saveRestoreOptions** { **SAVE**, **RESTORE_APPLICATION**, **RESTORE_-QEFRAMEWORK** }

Signals

- void **saveRestore** (SaveRestoreSignal::saveRestoreOptions option)

Public Member Functions

- void **setOwner** (PersistanceManager ∗ownerIn)
- void save ()
- void restore ()

9.128.1 Member Function Documentation

9.128.1.1 void SaveRestoreSignal::restore ( )

!! signal must be blocking

9.128.1.2 void SaveRestoreSignal::save ( )

!! signal must be blocking

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.129 saveRestoreSlot Class Reference

Public Slots

- void **saveRestore** (SaveRestoreSignal::saveRestoreOptions option)

Public Member Functions

- void **setOwner** (QEWidget ∗ownerIn)

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEWidget.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEWidget.cpp

9.130 selectMenu Class Reference

Public Member Functions

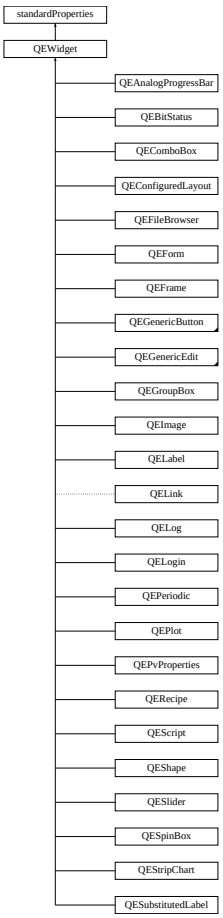
- **selectMenu** (QWidget ∗parent=0)
- imageContextMenu::imageContextMenuOptions **getSelectOption** (const QPoint &pos)
- void **setChecked** (const int mode)
- void **setPanEnabled** (bool enablePan)
- void **setVSliceEnabled** (bool enableVSliceSelection)
- void **setHSlicetEnabled** (bool enableHSliceSelection)
- void **setAreaEnabled** (bool enableAreaSelection)
- void **setProfileEnabled** (bool enableProfileSelection)
- void **setTargetEnabled** (bool enableTargetSelection)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/selectMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/selectMenu.cpp

9.131 standardProperties Class Reference

Inheritance diagram for standardProperties:



Public Member Functions

- **standardProperties** (QWidget ∗ownerIn)
- userLevelTypes::userLevels **getUserLevelVisibility** ()
- void **setUserLevelVisibility** (userLevelTypes::userLevels level)
- userLevelTypes::userLevels **getUserLevelEnabled** ()
- void **setUserLevelEnabled** (userLevelTypes::userLevels level)
- bool **getApplicationEnabled** () const
- void **setApplicationEnabled** (bool state)
- void **setRunVisible** (bool visibleIn)
- bool **getRunVisible** ()
- void **setDisplayAlarmState** (bool displayAlarmStateIn)
- bool **getDisplayAlarmState** ()

Protected Member Functions

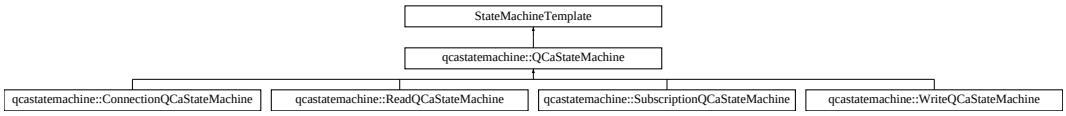
- void **checkVisibilityEnabledLevel** (userLevelTypes::userLevels level)

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

- /tmp/epicsqt/trunk/framework/widgets/include/standardProperties.h
- /tmp/epicsqt/trunk/framework/widgets/src/standardProperties.cpp

9.132 StateMachineTemplate Class Reference

Inheritance diagram for StateMachineTemplate:



Public Member Functions

- virtual bool **process** (int requestedState)=0

Public Attributes

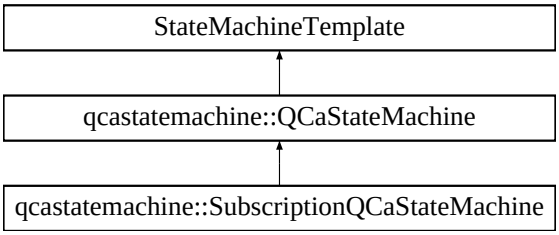
- int **currentState**
- int **requestState**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h

9.133 qcastatemachine::SubscriptionQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::SubscriptionQCaStateMachine:



Public Member Functions

- **SubscriptionQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.134 trace Class Reference

Public Attributes

- QVector< QCaDateTime > **timeStamps**
- QVector< double > **xdata**
- QVector< double > **ydata**
- QwtPlotCurve * **curve**
- QColor **color**
- QString **legend**
- bool **waveform**
- QwtPlotCurve::CurveStyle **style**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.h

9.135 TrackRange Class Reference

Public Member Functions

- void **clear** ()
- void **merge** (const double d)
- void **merge** (const TrackRange &that)
- bool **getMinMax** (double &min, double &max) const

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.cpp

9.136 userInfoStruct Class Reference

Public Attributes

- bool **enable**
- double **value1**
- double **value2**
- QString **elementText**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.137 QEPeriodic::userInfoStructArray Struct Reference

Public Attributes

- userInfoStruct **array** [NUM_ELEMENTS]

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.138 userLevelSignal Class Reference

Signals

- void userChanged (userLevelTypes::userLevels level)

  Internal use only. Send when the user level has changed.

Public Member Functions

- void **setLevel** (userLevelTypes::userLevels levelIn)
- userLevelTypes::userLevels **getLevel** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h
- /tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp

9.139 userLevelSlot Class Reference

Public Slots

- void **userChanged** (userLevelTypes::userLevels level)

Public Member Functions

- void **setOwner** (ContainerProfile *ownerIn)

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h
- /tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp

9.140 userLevelTypes Class Reference

Public Types

- enum userLevels { USERLEVEL_USER, USERLEVEL_SCIENTIST, USERLEVEL_ENGINEER }

9.140.1 Member Enumeration Documentation

9.140.1.1 enum userLevelTypes::userLevels

User levels set by widgets such as QELogin and used by many widgets to determine visibility, enabled state, and style.

Enumerator:

USERLEVEL_USER User level - least privilaged.

USERLEVEL_SCIENTIST User level - more privilaged than user, less than engineer.

USERLEVEL_ENGINEER User level - most privilaged.

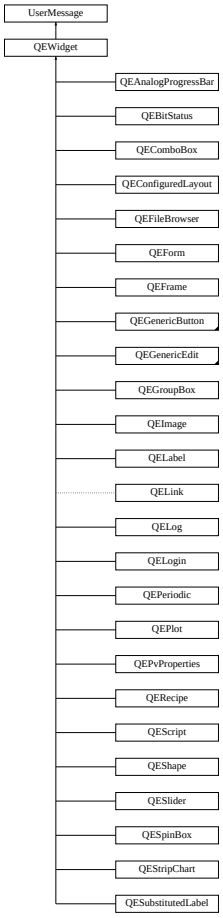
The documentation for this class was generated from the following file:

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.141 UserMessage Class Reference

```
#include <UserMessage.h>
```

Inheritance diagram for UserMessage:



Public Types

- enum **message_filter_options** { **MESSAGE_FILTER_ANY**, **MESSAGE_FILTER_-MATCH**, **MESSAGE_FILTER_NONE** }

Public Member Functions

- void setSourceId (unsigned int sourceId)

  Set the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets)
- void setFormId (unsigned int formId)

  Set the form ID (the the same ID for all sibling widgets within an QEForm widget)
- void setFormFilter (message_filter_options formFilterIn)

  Set the message filtering applied to the form ID.
- void setSourceFilter (message_filter_options sourceFilterIn)

Set the message filtering applied to the source ID.

- unsigned int getSourceId ()

  Get the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets.
- unsigned int getFormId ()

  Get the form ID (the the same ID for all sibling widgets within an QEForm widget)
- message_filter_options getFormFilter ()

  Get the message filtering applied to the form ID.
- message_filter_options getSourceFilter ()

  Get the message filtering applied to the source ID.
- void setChildFormId (unsigned int)

  Set the for ID of all widgets that are children of this widget.
- unsigned int getChildFormId ()

  Get the for ID of all widgets that are children of this widget.
- unsigned int getNextMessageFormId ()

  Generate a new form ID for all widgets in a new form.
- void sendMessage (QString message, message_types type=message_types(MESSAGE_TYPE_INFO))

  Send a message to the user.
- void sendMessage (QString message, QString source, message_types type=message_types(MESSAGE_TYPE_INFO))

  Send a message to the user with a source reference.
- QString getMessageTypeName (message_types type)

  Convenience function to provide string names for each message type.
- virtual void newMessage (QString, message_types)

  Virtual function to pass messages to derived classes (typicaly logging widgets or application windows)

Friends

- class UserMessageSlot
- class UserMessageSignal

9.141.1 Detailed Description

A class to manage user messages.

This class passes messages between widgets and application code

This class is used as a base class.

Messages are sent by calling sendMessage() Messages are received by implementing newMessage() in the derived class.

Messages can be filtered based on a source ID or a form ID

The derived widget is free to set the source ID to any value

Derived form widgets (QEForm) get a unique form ID using getNextMessageFormId() (as well as being able to set a source ID like any other QE widget) and pass this unique form ID to all widgets within the form using the ContainerProfile class.

Messages sent by a QE widget are received by all QE widgets and can filter the messages required by form ID and source ID. The form ID is under the management of the QEForm widget, the source ID is under the control of the GUI designer.

The QEForm widget does not display messages, but re-send them using its own form ID. Read on to see how this can be used.

Widgets that generate messages, and widgets (or application code) that use messages can be set up as follows:

- Application wide logging: An application with a single log window can can base a class on the UserMessage class and set up filtering to receive all messages. An application with log messages for seperate windows containing QEForm widgets (such as QEGui) can base each window class on the UserMessage class, then set up filtering for the appropriate form ID.

- Logging within a QEForm. A logging widget can be set to filter matching on the current form and so will pick up messages from any sibling widget. This includes messages from a sibling widget which is a nested QEForm. Whatever messages that nested form is set to receive, it will resend to its siblings. For example, if it is set to receive messages from the widgets it contains, these are resent up one level to the main form. If messages are dealt with within the nested QEForm (for example, it may have its own logging QE widget) then the nested QEForm could be set up not to filter and resend any messages.

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

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.142 UserMessageSignal Class Reference

`#include <UserMessage.h>`

Signals

- void message (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

  Emit a message signal. Any widget based on the UserMessage class can recieve these messages, filtered on formId and sourceId.

Public Member Functions

- void sendMessage (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

  Send a message to all widgets based on the UserMessage class.

9.142.1 Detailed Description

Class used to send message signals. Used only within UserMessage.cpp A single instance of this class is shared by all instances of the UserMessage class. This allows every UserMessage class instance to connect to a single source of messages

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

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.143 UserMessageSlot Class Reference

```
#include <UserMessage.h>
```

Public Slots

- void message (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

  A message has been received.

Public Member Functions

- void setOwner (UserMessage *ownerIn)

  Set the UserMessage class this is a part of.

9.143.1 Detailed Description

Class used to receive message signals. Used only within UserMessage.cpp An instance of this class is created by all instances of the UserMessage class. The UserMessage class uses an instance of this class to receive messages so it does not have to be based on QObject itself. This is required as derived classes generally need to be also based on another object derived from QObject (and QObject can only be the base of a single base class)

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

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.144 ValueScaling Class Reference

Public Member Functions

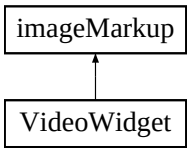
- void **reset** ()
- void **assign** (const ValueScaling &s)
- void **set** (const double dIn, const double mIn, const double cIn)
- void **get** (double &dOut, double &mOut, double &cOut)
- void **map** (const double fromLower, const double fromUpper, const double toLower, const double toUpper)
- bool **isScaled** ()
- double **value** (const double x)
- TrackRange **value** (const TrackRange &x)
- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.cpp

9.145 VideoWidget Class Reference

Inheritance diagram for VideoWidget:



Signals

- void **userSelection** (imageMarkup::markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)
- void **zoomInOut** (int zoomAmount)
- void **currentPixelInfo** (QPoint pos)
- void **pan** (QPoint pos)

Public Member Functions

- **VideoWidget** (QWidget *parent=0)
- void **setNewImage** (const QImage image, QCaDateTime &time)
- void **setPanning** (bool panningIn)

- bool **getPanning** ()
- QPoint **scalePoint** (QPoint pnt)
- int **scaleOrdinate** (int ord)
- QPoint **scaleImagePoint** (QPoint pnt)
- int **scaleImageOrdinate** (int ord)
- QImage **getImage** ()

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **mouseReleaseEvent** (QMouseEvent ∗event)
- void **mouseMoveEvent** (QMouseEvent ∗event)
- void **wheelEvent** (QWheelEvent ∗event)
- void **markupChange** (QImage &markups, QVector< QRect > &changedAreas)
- void **resizeEvent** (QResizeEvent ∗event)
- void **markupSetCursor** (QCursor cursor)
- void **markupAction** (markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/videowidget.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/videowidget.cpp

9.146 WidgetRef Class Reference

Public Member Functions

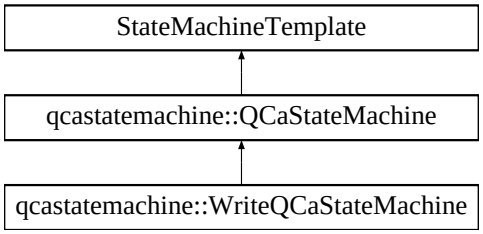
- **WidgetRef** (QEWidget ∗refIn)
- QEWidget ∗ **getRef** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.147 qcastatemachine::WriteQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::WriteQCaStateMachine:



Public Member Functions

- **WriteQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.148 zoomMenu Class Reference

Public Member Functions

- **zoomMenu** (QWidget ∗parent=0)
- void **enableAreaSelected** (bool enable)
- imageContextMenu::imageContextMenuOptions **getZoom** (const QPoint &pos)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/zoomMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/zoomMenu.cpp

Index