EPICS QT Framework

2.9.0

Generated by Doxygen 1.7.4

Wed Sep 10 2014 15:19:49

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

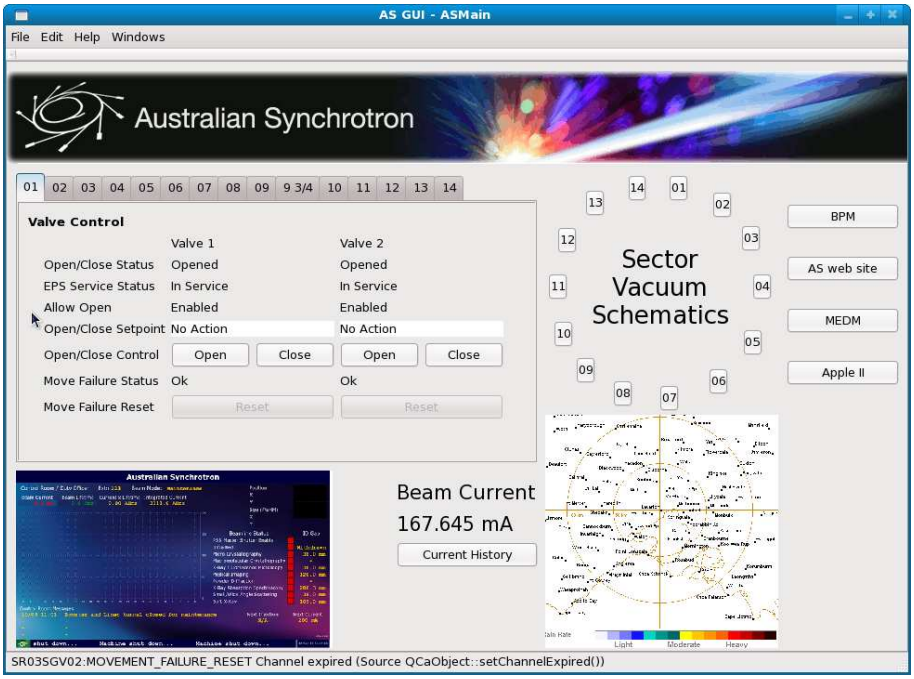


Figure 3.1: Australian Synchrotron mock up

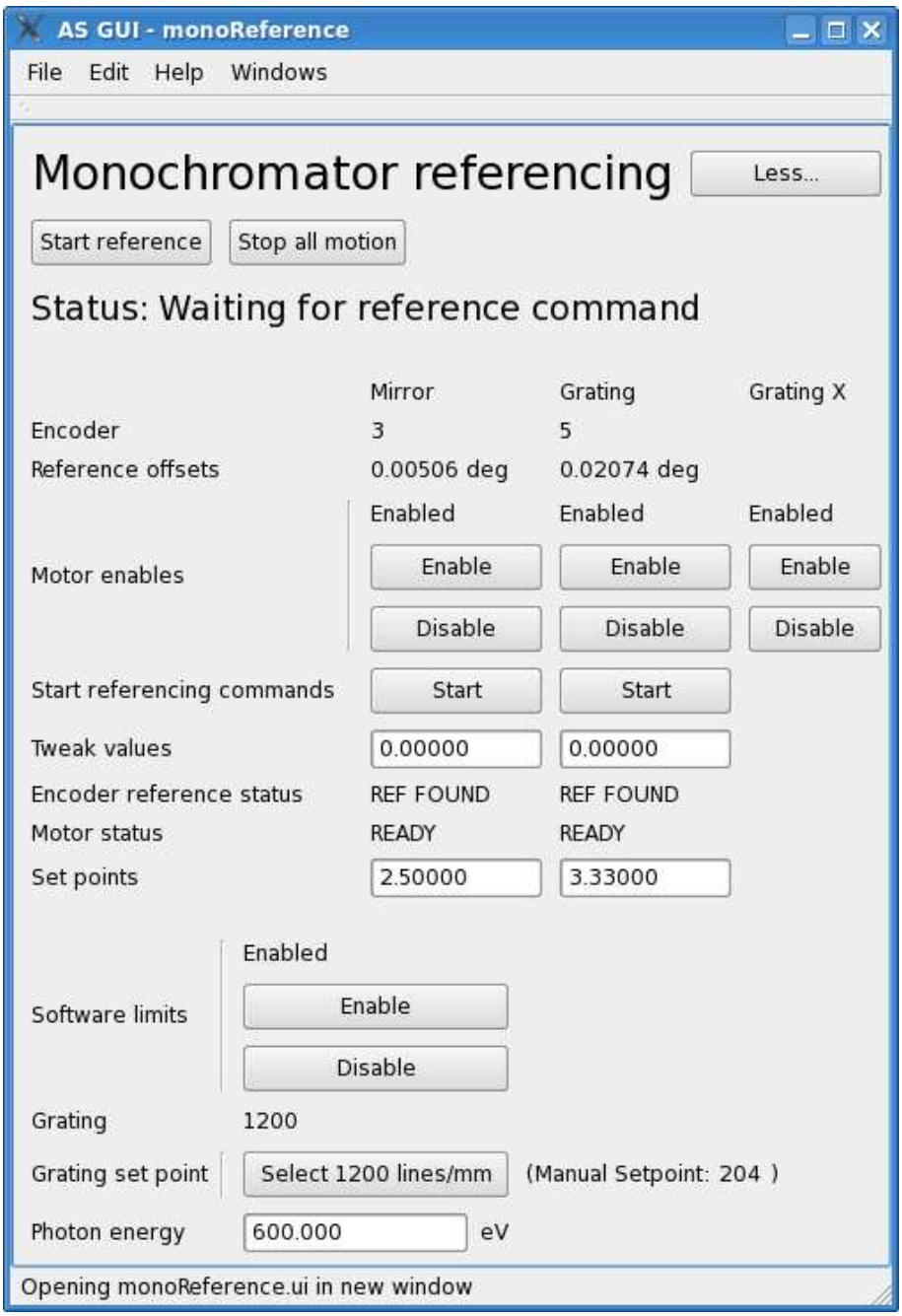


Figure 3.2: Monochromator referencing

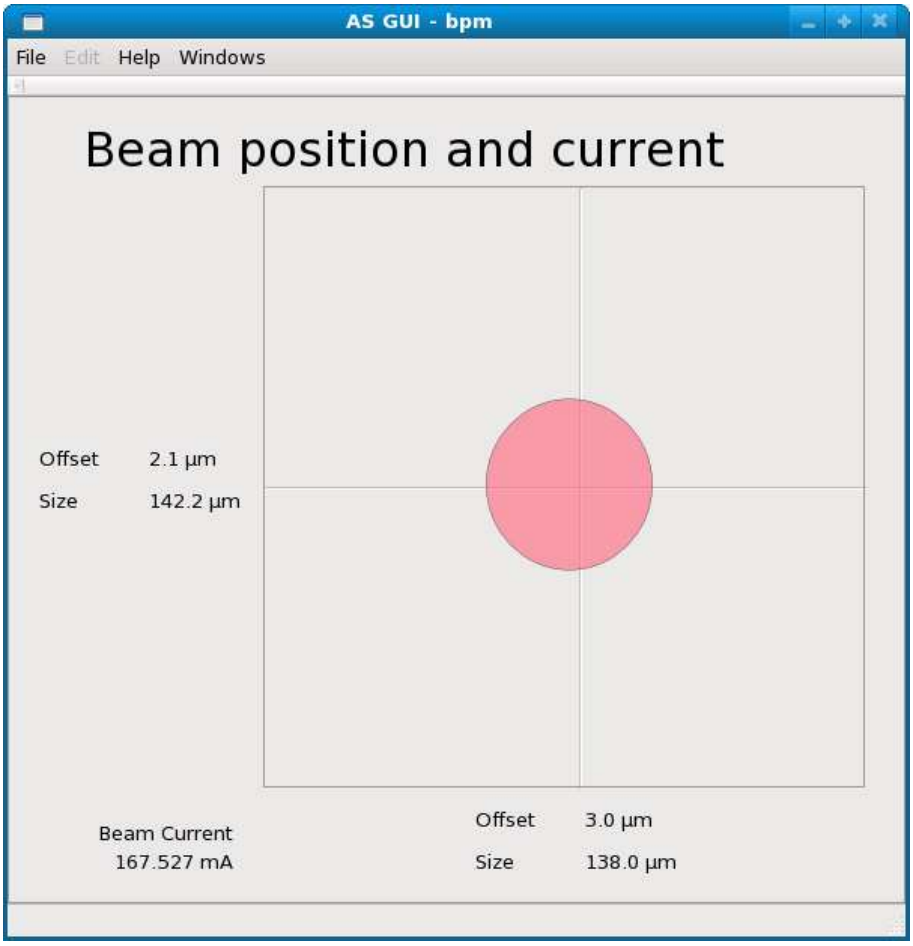


Figure 3.3: Beam position monitor

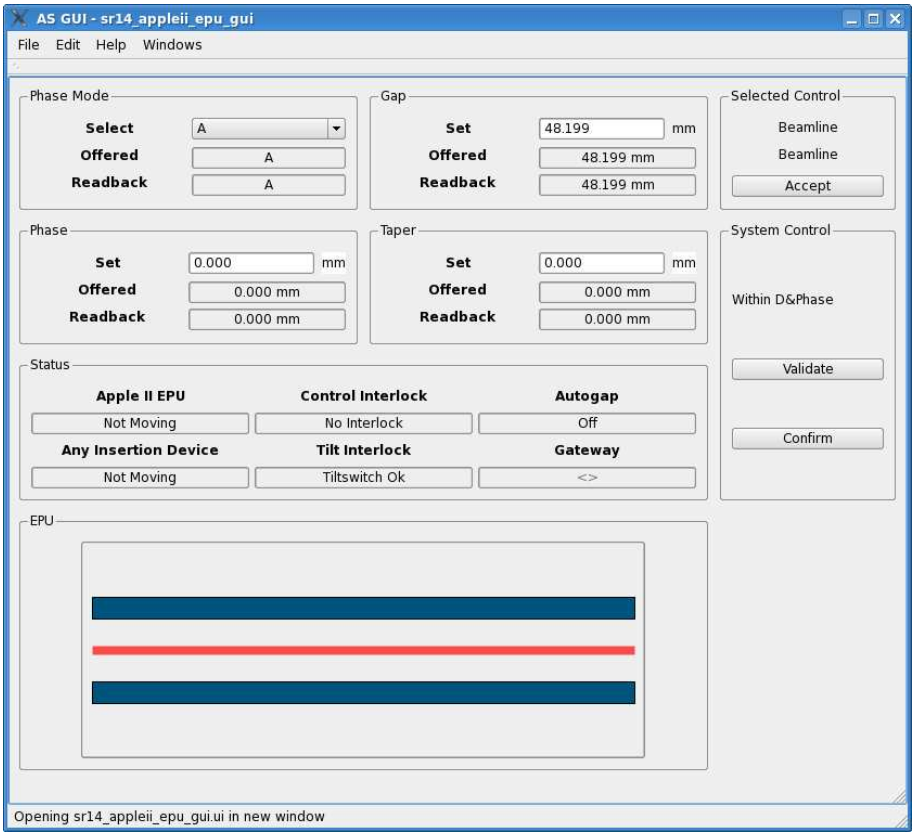


Figure 3.4: Insertion device

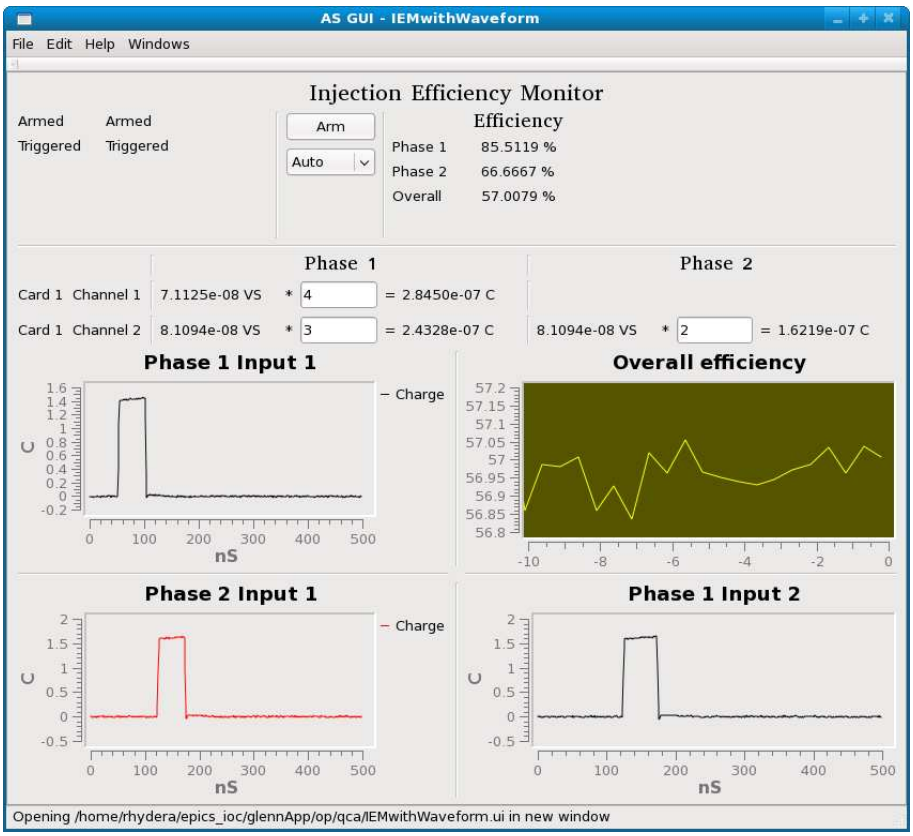


Figure 3.5: Injection efficiency monitor

Chapter 4

other applications using epicsqt widgets

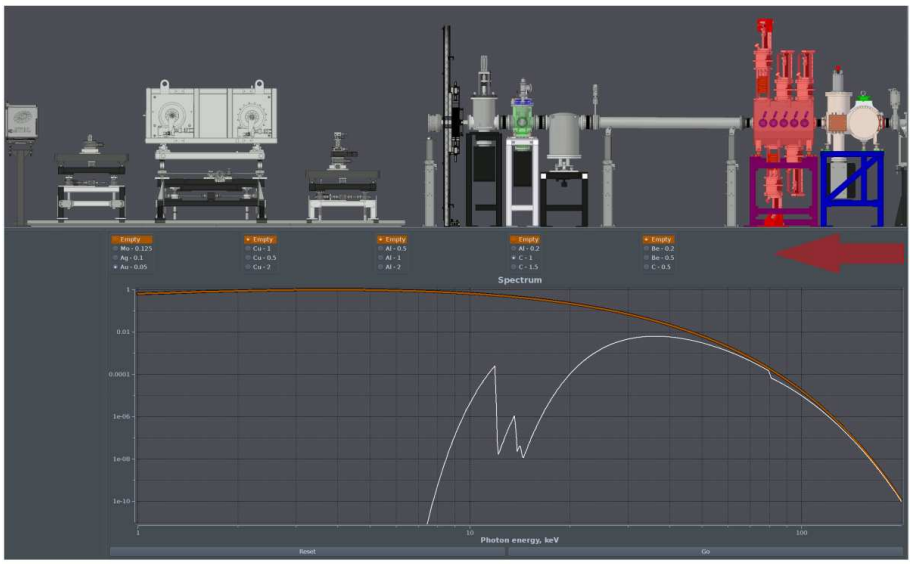


Figure 4.1: Medical Imaging beamline

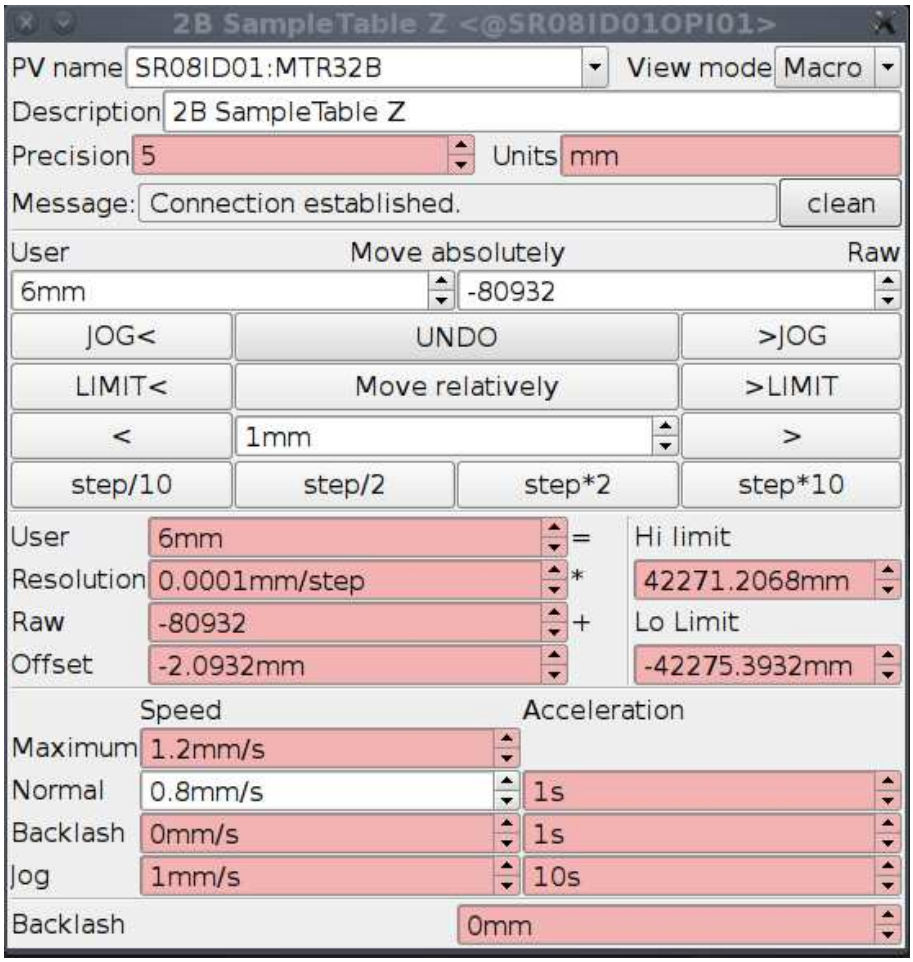


Figure 4.2: Motor controller

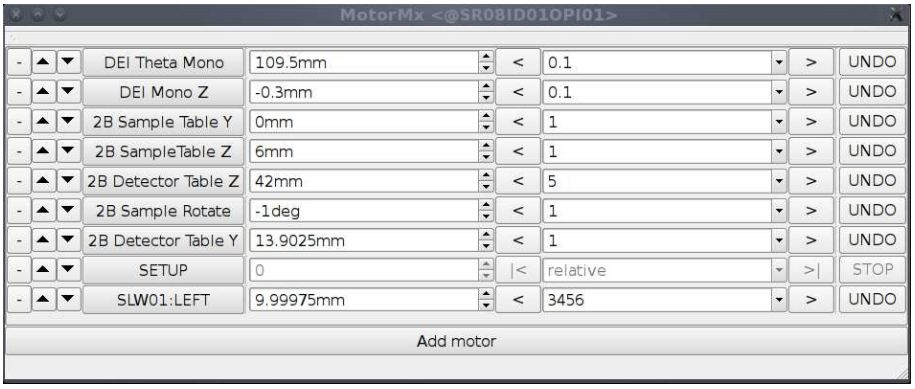


Figure 4.3: Motor controller

Chapter 5

Qt Designer

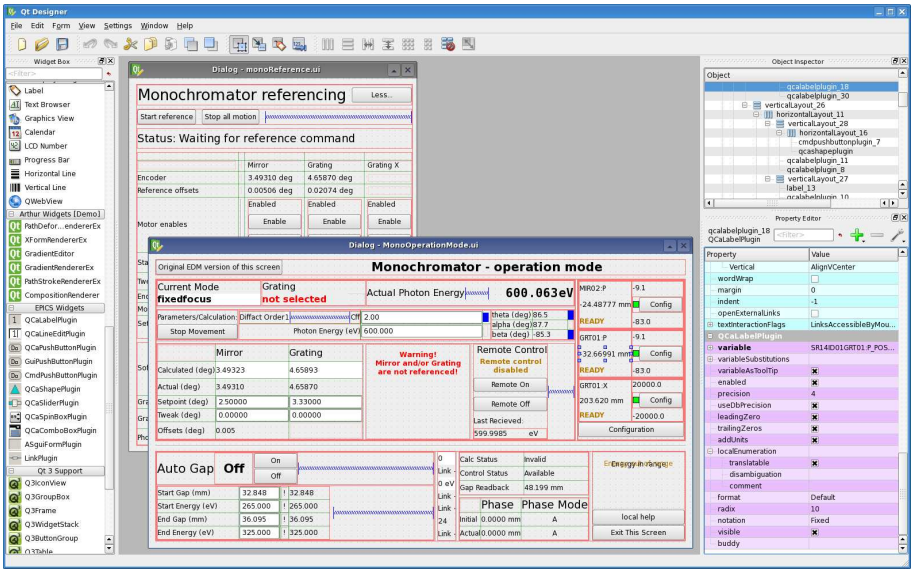


Figure 5.1: Editing multiple GUIs

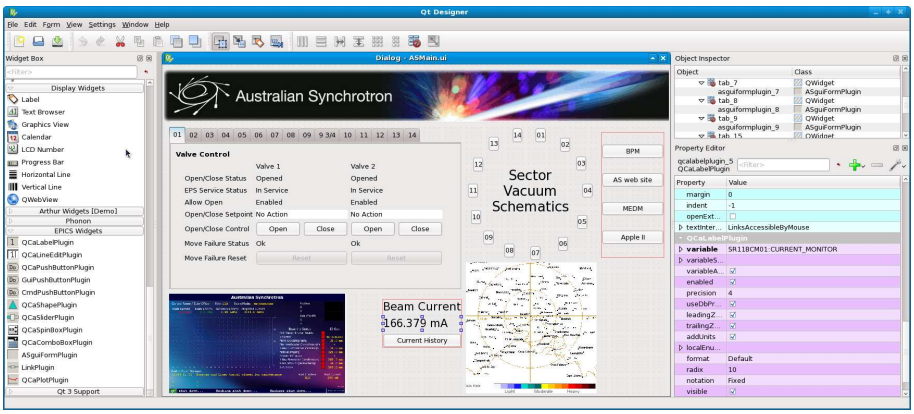


Figure 5.2: Editing a GUI

Chapter 6

Qt Creator

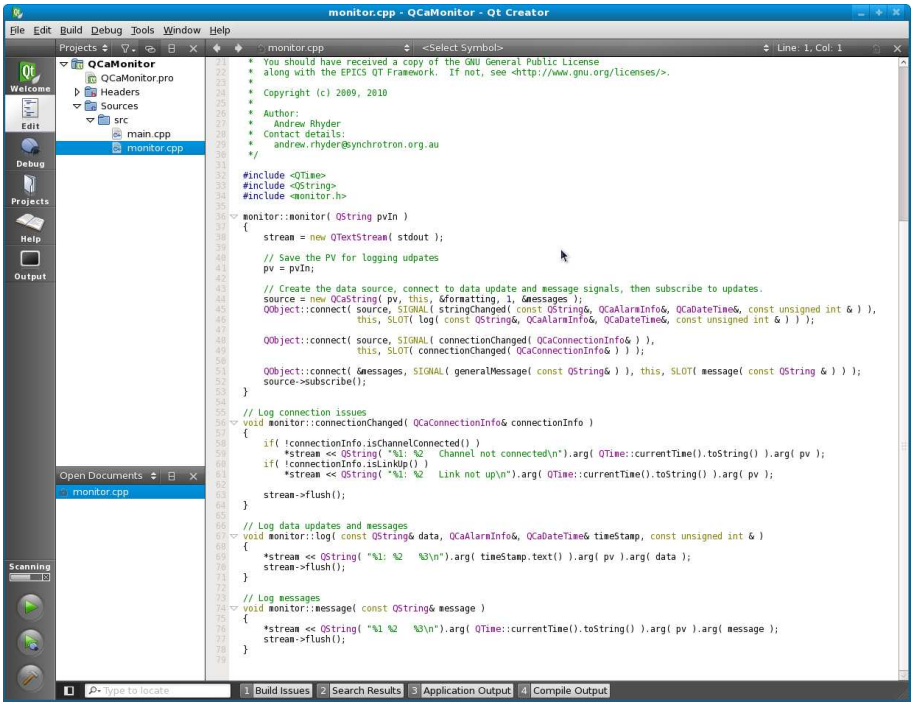


Figure 6.1: Application using epicsqt data source classes

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 _CopyPaste Class Reference

Public Member Functions

- **_CopyPaste** (bool pEnable, QString pProgram, QString pParameters, QString pWorkingDirectory, int pTimeOut, bool pStop, bool pLog)
- void **setEnable** (bool pEnable)
- bool **getEnable** ()
- void **setProgram** (QString pProgram)
- QString **getProgram** ()
- void **setParameters** (QString pParameters)
- QString **getParameters** ()
- void **setWorkingDirectory** (QString pWorkingDirectory)
- QString **getWorkingDirectory** ()
- void **setTimeOut** (int pTimeOut)
- int **getTimeOut** ()
- void **setStop** (bool pStop)
- bool **getStop** ()
- void **setLog** (bool pLog)
- bool **getLog** ()

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

9.2 _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 * **qeWidget**

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 _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

9.4 _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

9.5 _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

9.6 _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

9.7 _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

9.8 _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

9.9 areaInfo Class Reference

Public Member Functions

- void **setX1** (long x)
- void **setY1** (long y)
- void **setX2** (long x)
- void **setY2** (long y)
- void **setX** (long x)
- void **setY** (long y)
- void **setW** (long w)
- void **setH** (long h)
- void **setPoint1** (QPoint p1In)
- void **setPoint2** (QPoint p2In)
- void **clearX1** ()
- void **clearY1** ()

- void **clearX2** ()
- void **clearY2** ()
- void **clearX** ()
- void **clearY** ()
- void **clearW** ()
- void **clearH** ()
- bool **getStatus** ()
- QRect **getArea** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()

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

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

9.10 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

9.11 QEAnalogIndicator::BandList Class Reference

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

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

9.12 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

9.13 FFBuffer Class Reference

Public Member Functions

- void **reserve** ()
- void **release** ()
- bool **grabFree** ()

Public Attributes

- QMutex * **mutex**
- unsigned char * **mem**
- AVFrame * **pFrame**
- PixelFormat **pix_fmt**
- int **width**
- int **height**
- int **refs**

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

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

9.14 FFThread Class Reference

Public Slots

- void **stopGracefully** ()

Signals

- void **updateSignal** (FFBuffer *buf)

Public Member Functions

- **FFThread** (const QString &url, QObject ∗parent)
- void **run** ()

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

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

9.15 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

9.16 fullScreenWindow Class Reference

Signals

- void **fullScreenResize** ()

Public Member Functions

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

Protected Member Functions

- void **resizeEvent** (QResizeEvent ∗event)

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

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

9.17 histogram Class Reference

Public Member Functions

- **histogram** (QWidget ∗parent, imageDisplayProperties ∗idp)

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

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

9.18 histogramScroll Class Reference

Public Member Functions

- **histogramScroll** (QWidget ∗parent, imageDisplayProperties ∗idp)

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

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

9.19 historicImage Class Reference

Public Member Functions

- **historicImage** (QByteArray image, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &time)

Public Attributes

- QByteArray **image**
- unsigned long **dataSize**
- QCaAlarmInfo **alarmInfo**
- QCaDateTime **time**

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

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

9.20 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_ABOUT_IMAGE**, **ICM_ENABLE_VERT**, **ICM_ENABLE_HOZ**,

  ICM_ENABLE_AREA1, **ICM_ENABLE_AREA2**, **ICM_ENABLE_AREA3**, **ICM_-ENABLE_AREA4**,

  ICM_ENABLE_LINE, **ICM_ENABLE_TARGET**, **ICM_ENABLE_BEAM**, **ICM_DISPLAY_-BUTTON_BAR**,

  ICM_DISPLAY_IMAGE_DISPLAY_PROPERTIES, **ICM_DISPLAY_RECORDER**, **ICM_ZOOM_SELECTED**, **ICM_ZOOM_FIT**,

  ICM_ZOOM_PLUS, **ICM_ZOOM_MINUS**, **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**, **ICM_FULL_SCREEN**,

  ICM_DISPLAY_HSLICE, **ICM_DISPLAY_VSLICE**, **ICM_DISPLAY_AREA1**, **ICM_-DISPLAY_AREA2**,

  ICM_DISPLAY_AREA3, **ICM_DISPLAY_AREA4**, **ICM_DISPLAY_PROFILE**, **ICM_-DISPLAY_TARGET**,

  ICM_DISPLAY_BEAM, **ICM_DISPLAY_TIMESTAMP**, **ICM_DISPLAY_ELLIPSE**, **ICM_OPTIONS** }

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)

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

9.21 imageDisplayProperties Class Reference

Classes

- struct rgbPixel

Signals

- void **brightnessContrastAutoImage** ()
- void **imageDisplayPropertiesChange** ()

Public Member Functions

- void **setBrightnessContrast** (const unsigned int max, const unsigned int min)
- void **setAutoBrightnessContrast** (bool autoBrightnessContrast)
- void **setContrastReversal** (bool contrastReversal)
- void **setLog** (bool log)
- void **setFalseColour** (bool falseColour)
- bool **getAutoBrightnessContrast** ()
- bool **getContrastReversal** ()
- bool **getLog** ()
- bool **getFalseColour** ()
- int **getLowPixel** ()
- int **getHighPixel** ()
- void **setStatistics** (unsigned int minPIn, unsigned int maxPIn, unsigned int bitDepth, unsigned int binsIn[HISTOGRAM_BINS], rgbPixel pixelLookup[256])
- void **setHistZoom** (int value)
- int **getHistZoom** ()
- bool **statisticsValid** ()

Public Attributes

- int **zeroValue**
- int **fullValue**
- bool **defaultFullValue**
- unsigned int **range**
- unsigned int **maxP**
- unsigned int **minP**
- unsigned int **depth**
- unsigned int * **bins**
- bool **statisticsSet**

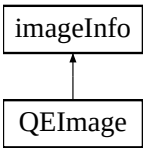
- rgbPixel * **pixelLookup**
- QLabel * **histXLabel**

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

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

9.22 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)
- void **infoUpdateZoom** ()
- void **infoUpdateZoom** (int value)
- void **infoUpdatePaused** ()
- void **infoUpdatePaused** (bool paused)
- void **setBriefInfoArea** (const bool briefIn)

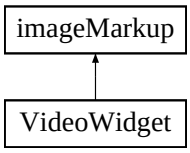
- bool **getBriefInfoArea** ()
- void **freshImage** (QDateTime &time)

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

9.23 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_ELLIPSE**, **MARKUP_-ID_COUNT**,

  MARKUP_ID_NONE }
- enum **beamAndTargetOptions** { **CROSSHAIR1**, **CROSSHAIR2** }

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, bool displayMarkups)
- void **markupHProfileChange** (int y, bool displayMarkups)
- void **markupVProfileChange** (int x, bool displayMarkups)

- void **markupLineProfileChange** (QPoint start, QPoint end, bool displayMarkups)

- void **markupTargetValueChange** (QPoint point, bool displayMarkups)
- void **markupBeamValueChange** (QPoint point, bool displayMarkups)
- void **markupEllipseValueChange** (QPoint point1, QPoint point2, bool displayMarkups)
- void **markupValueChange** (int markup, bool displayMarkups, QPoint p1, QPoint p2=QPoint())
- QCursor **getCircleCursor** ()
- QCursor **getTargetCursor** ()
- QCursor **getVLineCursor** ()
- QCursor **getHLineCursor** ()
- QCursor **getLineCursor** ()
- QCursor **getRegionCursor** ()
- virtual void **markupSetCursor** (QCursor cursor)=0
- void **setMarkupLegend** (markupIds mode, QString legend)
- QString **getMarkupLegend** (markupIds mode)
- void **clearMarkup** (markupIds markupId)
- void **showMarkup** (markupIds markupId)
- void **displayMarkup** (markupIds markupId, bool state)
- bool **isMarkupVisible** (markupIds mode)
- double **getZoomScale** ()
- QSize **getImageSize** ()
- void **setImageSize** (const QSize &imageSizeIn)
- beamAndTargetOptions **getTargetOption** ()
- void **setTargetOption** (beamAndTargetOptions option)
- beamAndTargetOptions **getBeamOption** ()
- void **setBeamOption** (beamAndTargetOptions option)
- void **setBeamOrTargetOption** (markupIds item, beamAndTargetOptions option)

Public Attributes

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

Protected Member Functions

- void **drawMarkups** (QPainter &p, const QRect &rect)
- bool **anyVisibleMarkups** ()
- 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** (const double scale)
- virtual void **markupChange** (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

9.24 imageUpdateIndicator Class Reference

Public Member Functions

- void **freshImage** ()
- void **paintEvent** (QPaintEvent ∗)

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

9.25 loginWidget Class Reference

Public Member Functions

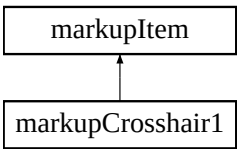
- **loginWidget** (QELogin ∗ownerIn)
- userLevelTypes::userLevels **getUserType** ()
- QString **getPassword** ()
- void **clearPassword** ()

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

9.26 markupCrosshair1 Class Reference

Inheritance diagram for markupCrosshair1:



Public Member Functions

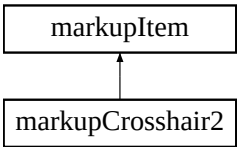
- **markupCrosshair1** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

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

9.27 markupCrosshair2 Class Reference

Inheritance diagram for markupCrosshair2:



Public Member Functions

- **markupCrosshair2** (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** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

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

9.28 markupDisplayMenu Class Reference

Public Member Functions

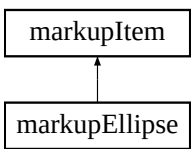
- **markupDisplayMenu** (QWidget ∗parent=0)
- void **setDisplayed** (imageContextMenu::imageContextMenuOptions option, bool state)
- void **setItemText** (imageContextMenu::imageContextMenuOptions option, QString title)
- bool **isDisplayed** (imageContextMenu::imageContextMenuOptions option)
- void **enable** (imageContextMenu::imageContextMenuOptions option, bool state)

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

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

9.29 markupEllipse Class Reference

Inheritance diagram for markupEllipse:



Public Member Functions

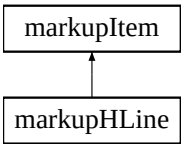
- **markupEllipse** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

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

9.30 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** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

9.30.1 Member Function Documentation

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

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

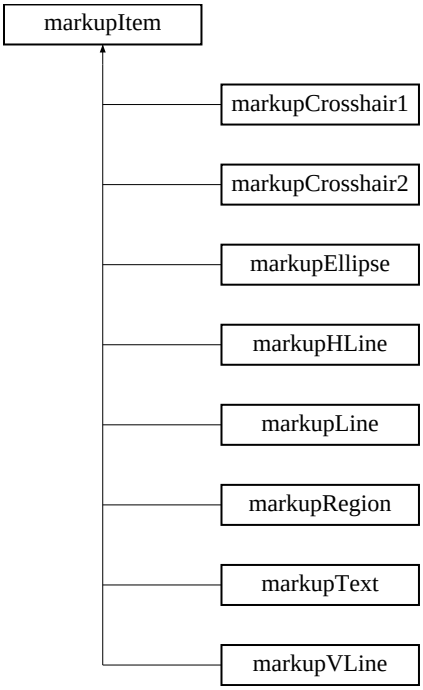
Implements markupItem.

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

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

9.31 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 **drawMarkupItem** (QPainter &p)
- void **scale** (const double xScale, const double yScale, const double zoomScale)
- QSize **getImageSize** ()
- 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 QCursor **defaultCursor** ()=0
- virtual void **nonInteractiveUpdate** (QPoint, QPoint)
- void **setThickness** (const unsigned int thicknessIn)
- unsigned int **getThickness** ()
- void **setLegend** (const QString legendIn)
- const QString **getLegend** ()
- void **setColor** (QColor colorIn)
- QColor **getColor** ()

Public Attributes

- QRect **area**
- QRect **scalableArea**
- 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)
- const QSize **getLegendSize** ()
- void **addLegendArea** ()
- const QPoint **getLegendTextOrigin** (QPoint posScaled)
- void **setLegendOffset** (QPoint offset, legendJustification just)
- const QPoint **getLegendOffset** ()
- void **drawLegend** (QPainter &p, QPoint posScaled)
- QPoint **limitPointToImage** (const QPoint pos)
- double **getZoomScale** ()

Protected Attributes

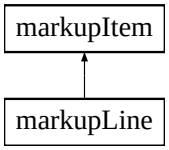
- markupHandles **activeHandle**
- imageMarkup ∗ **owner**
- unsigned int **thickness**
- unsigned int **maxThickness**

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

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

9.32 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** ()

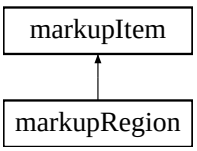
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

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

9.33 markupRegion Class Reference

Inheritance diagram for markupRegion:



Public Member Functions

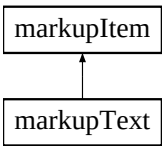
- **markupRegion** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

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

9.34 markupText Class Reference

Inheritance diagram for markupText:



Public Member Functions

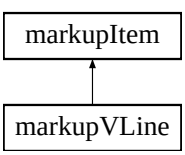
- **markupText** (imageMarkup ∗ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **setText** (QString textIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

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

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

9.35 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** ()
- QCursor **defaultCursor** ()
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

9.35.1 Member Function Documentation

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

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

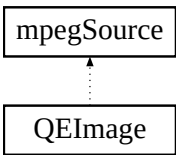
Implements markupItem.

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

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

9.36 mpegSource Class Reference

Inheritance diagram for mpegSource:



Public Member Functions

- void updateImage (FFBuffer ∗buf)
- void **setURL** (QString)
- void **startStream** ()
- void **stopStream** ()

Protected Member Functions

- QString **getURL** ()
- void **setURL** (QString urlIn)
- void **stopStream** ()
- void **startStream** ()

9.36.1 Member Function Documentation

9.36.1.1 void mpegSource::updateImage (FFBuffer * *buf*)

!!??? * 3 for color only

!! Since the QEImage widget handles (or should handle) CA image data in all the formats that are expected in this mpeg stream !! perhaps this formatting here should be simply packaging the data in a QbyteArray and delivering it, rather than perform any conversion.

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

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

9.37 mpegSourceObject Class Reference

Public Slots

- void **updateImage** (FFBuffer *buf)

Signals

- void **aboutToQuit** ()

Public Member Functions

- **mpegSourceObject** (mpegSource *msIn)
- void **sentAboutToQuit** ()

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

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

9.38 QEStripChartToolBar::OwnWidgets Class Reference

Public Member Functions

- **OwnWidgets** (QEStripChartToolBar ∗parent)

Public Attributes

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

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

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

9.39 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

9.40 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

9.41 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

9.42 playbackTimer Class Reference

Public Member Functions

- **playbackTimer** (recording *recorderIn)
- void **timerEvent** (QTimerEvent *event)

Public Attributes

- recording * **recorder**

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

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

9.43 pointInfo Class Reference

Public Member Functions

- void **setX** (long x)
- void **setY** (long y)
- void **setPoint** (QPoint pIn)
- void **clearX** ()
- void **clearY** ()
- bool **getStatus** ()
- QPoint **getPoint** ()

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

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

9.44 profilePlot Class Reference

Public Types

- enum **plotDirections** { **PROFILEPLOT_LR**, **PROFILEPLOT_RL**, **PROFILEPLOT_TB**, **PROFILEPLOT_BT** }

Public Member Functions

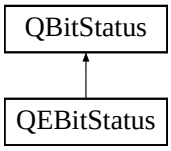
- **profilePlot** (plotDirections plotDirectionIn)
- void **setProfile** (QVector< QPointF > *profile, double minX, double maxX, double minY, double maxY, QString title, QPoint start, QPoint end, unsigned int thicknessIn)
- void **clearProfile** ()

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

9.45 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** ()

Protected Member Functions

- void **setIsActive** (const bool value)
- bool **getIsActive** ()

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**
- bool **isActive**

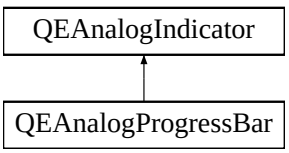
The documentation for this class was generated from the following files:

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

9.46 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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

  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 () const

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

Protected Member Functions

- virtual QString **getTextImage** ()
- virtual BandList **getBandList** ()
- void **setIsActive** (const bool value)
- bool **getIsActive** () const

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
- bool isActive

  Alternative to isEnabled. Default is true.

9.46.1 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.

9.46.2 Member Enumeration Documentation

9.46.2.1 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)

9.46.2.2 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.

9.46.3 Property Documentation

9.46.3.1 QColor QEAnalogIndicator::backgroundColour [read, write]

Background colour

9.46.3.2 QColor QEAnalogIndicator::borderColour [read, write]

Border colour

9.46.3.3 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.

9.46.3.4 QColor QEAnalogIndicator::fontColour [read, write]

Font colour

9.46.3.5 QColor QEAnalogIndicator::foregroundColour [read, write]

Foreground colour

9.46.3.6 bool QEAnalogIndicator::logScale [read, write]

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

9.46.3.7 int QEAnalogIndicator::logScaleInterval [read, write]

Log scale interval.

9.46.3.8 double QEAnalogIndicator::majorInterval [read, write]

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

9.46.3.9 double QEAnalogIndicator::maximum [read, write]

Maximum indicated value.

9.46.3.10 double QEAnalogIndicator::minimum [read, write]

Minimum indicated value.

9.46.3.11 double QEAnalogIndicator::minorInterval [read, write]

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

9.46.3.12 Modes QEAnalogIndicator::mode [read, write]

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

9.46.3.13 Orientations QEAnalogIndicator::orientation [read, write]

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

9.46.3.14 bool QEAnalogIndicator::showScale [read, write]

If set, show the scale

9.46.3.15 bool QEAnalogIndicator::showText [read, write]

If set, show textual representation of value on the indicator

9.46.3.16 int QEAnalogIndicator::spanAngle [read, write]

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

9.46.3.17 double QEAnalogIndicator::value [read, write]

Current indicated value.

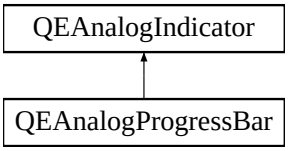
Reimplemented in QEAnalogProgressBar.

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

9.47 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, 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 }

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 value
- bool isActive

  Alternative to isEnabled. Default is true.
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction

9.47.1 Member Enumeration Documentation

9.47.1.1 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.

9.47.1.2 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.

9.47.1.3 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.

9.47.1.4 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.

9.47.2 Constructor & Destructor Documentation

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

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

9.47.2.2 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.

9.47.3 Member Function Documentation

9.47.3.1 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.

9.47.4 Property Documentation

9.47.4.1 bool QEAnalogProgressBar::addUnits [read, write]

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

9.47.4.2 AlarmSeverityDisplayModes QEAnalogProgressBar::alarmSeverityDisplayMode [read, write]

Visualise the EPICS alarm severity

9.47.4.3 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.

9.47.4.4 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'.

9.47.4.5 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.

9.47.4.6 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).

9.47.4.7 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.

9.47.4.8 bool QEAnalogProgressBar::leadingZero [read, write]

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

9.47.4.9 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"

9.47.4.10 Notations QEAnalogProgressBar::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.47.4.11 int QEAnalogProgressBar::precision [read, write]

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

9.47.4.12 bool QEAnalogProgressBar::trailingZeros [read, write]

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

9.47.4.13 bool QEAnalogProgressBar::useDbDisplayLimits [read, write]

Use the EPICS database display limits

9.47.4.14 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.

9.47.4.15 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'.

9.47.4.16 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.

9.47.4.17 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.

9.47.4.18 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.

9.47.4.19 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'.

9.47.4.20 int QEAnalogProgressBar::value [read, write]

Current indicated value.

Reimplemented from QEAnalogIndicator.

9.47.4.21 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

9.47.4.22 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.

9.47.4.23 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.

9.47.4.24 bool QEAnalogProgressBar::visible [read, write]

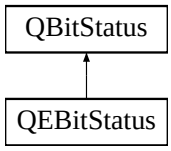
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.48 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
- double **value**
- bool **isActive**

9.48.1 Member Enumeration Documentation

9.48.1.1 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.

9.48.2 Member Function Documentation

9.48.2.1 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.

9.48.3 Property Documentation

9.48.3.1 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.

9.48.3.2 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.

9.48.3.3 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.

9.48.3.4 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'.

9.48.3.5 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.

9.48.3.6 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.

9.48.3.7 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.

9.48.3.8 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'.

9.48.3.9 QString QEBitStatus::variable [read, write]

EPICS variable name (CA PV)

9.48.3.10 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.

9.48.3.11 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.

9.48.3.12 bool QEBitStatus::visible [read, write]

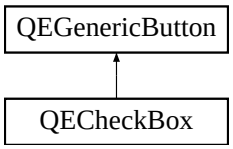
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.49 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 ProgramStartupOptionNames { None = applicationLauncher::PSO_NONE, Terminal = applicationLauncher::PSO_TERMINAL, LogOutput = applicationLauncher::PSO_LOGOUTPUT, StdOutput = applicationLauncher::PSO_STDOUTPUT }
- enum CreationOptionNames {

  Open = QEActionRequests::OptionOpen, NewTab = QEActionRequests::OptionNewTab, NewWindow = QEActionRequests::OptionNewWindow, DockTop = QEActionRequests::OptionTopDockWindow,

  DockBottom = QEActionRequests::OptionBottomDockWindow, DockLeft = QEActionRequests::OptionLeftDockWindow, DockRight = QEActionRequests::OptionRightDockWindow, DockTopTabbed = QEActionRequests::OptionTopDockWindowTabbed,

DockBottomTabbed = QEActionRequests::OptionBottomDockWindowTabbed, DockLeftTabbed = QEActionRequests::OptionLeftDockWindowTabbed, DockRightTabbed = QEActionRequests::OptionRightDockWindowTabbed, DockFloating = QEActionRequests::OptionFloatingDockWindow }

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 requestAction (const QEActionRequests &request)

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 (const QEActionRequests &request)

  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)
- void programCompleted ()

  Program started by button has compelted.

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

9.49.1 Member Enumeration Documentation

9.49.1.1 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.

9.49.1.2 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.

DockTop Open new GUI in a top dock window.

DockBottom Open new GUI in a bottom dock window.

DockLeft Open new GUI in a left dock window.

DockRight Open new GUI in a right dock window.

DockTopTabbed Open new GUI in a top dock window (tabbed with any existing dock in that area)

DockBottomTabbed Open new GUI in a bottom dock window (tabbed with any existing dock in that area)

DockLeftTabbed Open new GUI in a left dock window (tabbed with any existing dock in that area)

DockRightTabbed Open new GUI in a right dock window (tabbed with any existing dock in that area)

DockFloating Open new GUI in a floating dock window.

9.49.1.3 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.

9.49.1.4 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.

9.49.1.5 enum QECheckBox::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a termainal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send doutput to standard output and standard error.

9.49.1.6 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)

9.49.1.7 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.

9.49.2 Constructor & Destructor Documentation

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

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

9.49.2.2 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.

9.49.3 Member Function Documentation

9.49.3.1 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)

9.49.3.2 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.

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

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

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

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

9.49.3.5 void QECheckBox::requestAction (const QEActionRequests & *request*) [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.

9.49.4 Property Documentation

9.49.4.1 bool QECheckBox::addUnits [read, write]

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

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

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

9.49.4.3 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.

9.49.4.4 QStringList QECheckBox::arguments [read, write]

Arguments for program specified in the 'program' property.

9.49.4.5 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'.

9.49.4.6 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.

9.49.4.7 QString QECheckBox::clickText [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.8 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

9.49.4.9 QString QECheckBox::confirmText [read, write]

Text used to confirm acion if confirmation dialog is presented

Reimplemented from QEGenericButton.

9.49.4.10 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.

9.49.4.11 QString QECheckBox::customisationName [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEGui can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI.

Reimplemented from QEGenericButton.

9.49.4.12 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.

9.49.4.13 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).

9.49.4.14 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.

9.49.4.15 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.

9.49.4.16 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.

9.49.4.17 bool QECheckBox::leadingZero [read, write]

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

9.49.4.18 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"

9.49.4.19 Notations QECheckBox::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.49.4.20 QString QECheckBox::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.49.4.21 QPixmap QECheckBox::pixmap0 [read, write]

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

9.49.4.22 QPixmap QECheckBox::pixmap1 [read, write]

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

9.49.4.23 QPixmap QECheckBox::pixmap2 [read, write]

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

9.49.4.24 QPixmap QECheckBox::pixmap3 [read, write]

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

9.49.4.25 QPixmap QECheckBox::pixmap4 [read, write]

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

9.49.4.26 QPixmap QECheckBox::pixmap5 [read, write]

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

9.49.4.27 QPixmap QECheckBox::pixmap6 [read, write]

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

9.49.4.28 QPixmap QECheckBox::pixmap7 [read, write]

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

9.49.4.29 int QECheckBox::precision [read, write]

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

9.49.4.30 QString QECheckBox::pressText [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.31 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 substittions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from QEGenericButton.

9.49.4.32 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

9.49.4.33 ProgramStartupOptionNames QECheckBox::programStartupOption [read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.49.4.34 QString QECheckBox::releaseText [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.35 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)

9.49.4.36 bool QECheckBox::trailingZeros [read, write]

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

9.49.4.37 UpdateOptions QECheckBox::updateOption [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.38 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.

9.49.4.39 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'.

9.49.4.40 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.

9.49.4.41 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.

9.49.4.42 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.

9.49.4.43 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'.

9.49.4.44 QString QECheckBox::variable [read, write]

EPICS variable name (CA PV)

9.49.4.45 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.

9.49.4.46 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.

9.49.4.47 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.

9.49.4.48 bool QECheckBox::writeOnClick [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.49 bool QECheckBox::writeOnPress [read, write]

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

Reimplemented from QEGenericButton.

9.49.4.50 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

9.50 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

9.51 QEComboBox Class Reference

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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

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

9.51.1 Member Enumeration Documentation

9.51.1.1 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.

9.51.2 Member Function Documentation

9.51.2.1 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.

9.51.3 Member Data Documentation

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

Use database enumerations - defaults to true

9.51.3.2 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).

9.51.4 Property Documentation

9.51.4.1 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.

9.51.4.2 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.

9.51.4.3 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.

9.51.4.4 QString QEComboBox::localEnumeration [read, write]

Enumrations values used when useDbEnumerations is false.

9.51.4.5 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)

9.51.4.6 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'.

9.51.4.7 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.

9.51.4.8 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.

9.51.4.9 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.

9.51.4.10 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'.

9.51.4.11 QString QEComboBox::variable [read, write]

EPICS variable name (CA PV)

9.51.4.12 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.

9.51.4.13 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.

9.51.4.14 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

9.52 QEConfiguredLayout Class Reference

Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum UserLevels { 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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()

- void **refreshFields** ()
- void **userLevelChanged** (userLevelTypes::userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)

- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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.

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 **optionsLayout**
- int **currentUserType**
- bool **subscription**

Properties

- QString **itemDescription**
- bool **showItemList**
- configurationTypesProperty **configurationType**
- optionsLayoutProperty optionsLayout

  Change the order of the widgets. Valid orders are: TOP, BOTTOM, LEFT and RIG.

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

9.52.1 Member Enumeration Documentation

9.52.1.1 enum QEConfiguredLayout::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.

9.52.2 Property Documentation

9.52.2.1 bool QEConfiguredLayout::allowDrop [read, write]

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

9.52.2.2 bool QEConfiguredLayout::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.

9.52.2.3 unsigned QEConfiguredLayout::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.

9.52.2.4 UserLevels QEConfiguredLayout::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'.

9.52.2.5 QString QEConfiguredLayout::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.

9.52.2.6 QString QEConfiguredLayout::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.

9.52.2.7 QString QEConfiguredLayout::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.

9.52.2.8 UserLevels QEConfiguredLayout::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'.

9.52.2.9 bool QEConfiguredLayout::variableAsToolTip [read, write]

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

9.52.2.10 bool QEConfiguredLayout::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/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.53 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

9.54 QEFileBrowser Class Reference

```
#include <QEFileBrowser.h>
```

Public Types

- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void selected (QString pFilename)

Public Member Functions

- **QEFileBrowser** (QWidget *pParent=0)
- void **setVariableName** (QString pValue)
- QString **getVariableName** ()
- void **setVariableNameSubstitutions** (QString pValue)
- QString **getVariableNameSubstitutions** ()
- 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 **setShowTable** (bool pValue)
- bool **getShowTable** ()
- 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 **setFileDialogDirectoriesOnly** (bool pValue)
- bool **getFileDialogDirectoriesOnly** ()
- void **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **updateTable** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 Attributes

- QELineEdit * **qeLineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetFileBrowser * **qTableWidgetFileBrowser**
- QString fileFilter

  *Specify which files to browse. To specify more than one filter, please separate them with a ";". Example: *.py;*.ui (this will only display files with an extension .py or .ui).*

- bool showFileExtension

  Show/hide the extension of files.

- bool fileDialogDirectoriesOnly

  Enable/disable the browsing of directories-only when opening the dialog window.

- int **optionsLayout**

Properties

- QString variable
- QString variableSubstitutions
- QString directoryPath

  Default directory where to browse files when QEFileBrowser is launched for the first time.

- bool showDirectoryPath

  Show/hide directory path line edit where the user can specify the directory to browse files.

- bool showDirectoryBrowser

  Show/hide button to open the dialog window to browse for directories and files.

- bool showRefresh

  Show/hide button to refresh the table containing the list of files being browsed.

- bool showTable

  Show/hide table containing the list of files being browsed.

- bool showColumnTime

Show/hide column containing the time of creation of files.

- bool showColumnSize

  Show/hide column containing the size (in bytes) of files.

- bool showColumnFilename

  Show/hide column containing the name of files.

- optionsLayoutProperty optionsLayout

  Change the order of the widgets. Valid orders are: TOP, BOTTOM, LEFT and RIG.

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

9.54.1 Detailed Description

This class is a EPICS aware widget. The QEFileBrowser widget allows the user to browse existing files from a certain directory.

9.54.2 Member Enumeration Documentation

9.54.2.1 enum QEFileBrowser::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.

9.54.3 Member Function Documentation

9.54.3.1 void QEFileBrowser::selected (QString *pFilename*) [signal]

Signal that is generated every time the user double-clicks a certain file. This signals emits a string that contains the full path and the name of the selected file. This signal may be captured by other widgets that perform further operations (for instance, the QEImage displays the content of this file if it is a graphical one).

9.54.4 Property Documentation

9.54.4.1 bool QEFileBrowser::allowDrop [read, write]

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

9.54.4.2 bool QEFileBrowser::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.

9.54.4.3 unsigned QEFileBrowser::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.

9.54.4.4 UserLevels QEFileBrowser::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'.

9.54.4.5 QString QEFileBrowser::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.

9.54.4.6 QString QEFileBrowser::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.

9.54.4.7 QString QEFileBrowser::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.

9.54.4.8 UserLevels QEFileBrowser::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 setUser-Level() 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'.

9.54.4.9 QString QEFileBrowser::variable [read, write]

EPICS variable name (CA PV). This variable is used for both writing and reading the directory to be used by the widget.

9.54.4.10 bool QEFileBrowser::variableAsToolTip [read, write]

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

9.54.4.11 QString QEFileBrowser::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.

9.54.4.12 bool QEFileBrowser::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/QEFileBrowser/QEFileBrowser.h
- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.55 QEForm Class Reference

Public Types

- enum **MessageFilterOptions** { **Match** = UserMessage::MESSAGE_FILTER_-MATCH, **None** = UserMessage::MESSAGE_FILTER_NONE }

Public Slots

- bool **readUiFile** ()

Public Member Functions

- **QEForm** (QWidget ∗parent=0)
- **QEForm** (const QString &uifileNameIn, QWidget ∗parent=0)
- void **commonInit** (const bool alertIfUINoFoundIn, const bool loadManuallyIn)
- void **setQEGuiTitle** (const QString titleIn)
- QString **getQEGuiTitle** ()
- QString **getFullFileName** ()
- QString **getUiFileName** ()
- void **setFileMonitoringIsEnabled** (bool fileMonitoringIsEnabled)
- bool **getFileMonitoringIsEnabled** ()
- void **setHandleGuiLaunchRequests** (bool handleGuiLaunchRequests)
- bool **getHandleGuiLaunchRequests** ()
- void **setResizeContents** (bool resizeContentsIn)
- bool **getResizeContents** ()
- QString **getContainedFrameworkVersion** ()
- QString **getUniqueIdentifier** ()
- void **setUniqueIdentifier** (QString name)
- int **getDisconnectedCount** ()
- int **getConnectedCount** ()
- void **setUiFileNameProperty** (QString uiFileName)
- QString **getUiFileNameProperty** ()
- void **setVariableNameSubstitutionsProperty** (QString variableNameSubstitutions)
- QString **getVariableNameSubstitutionsProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Attributes

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

Properties

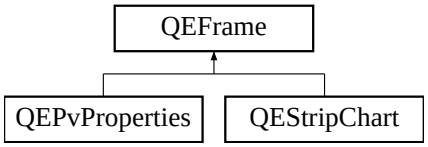
- QString **uiFile**
- QString **variableSubstitutions**
- unsigned **int**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**

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

9.56 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

9.56.1 Member Enumeration Documentation

9.56.1.1 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.

9.56.2 Property Documentation

9.56.2.1 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.

9.56.2.2 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.

9.56.2.3 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.

9.56.2.4 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'.

9.56.2.5 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.

9.56.2.6 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.

9.56.2.7 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.

9.56.2.8 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'.

9.56.2.9 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.

9.56.2.10 bool QEFrame::visible [read, write]

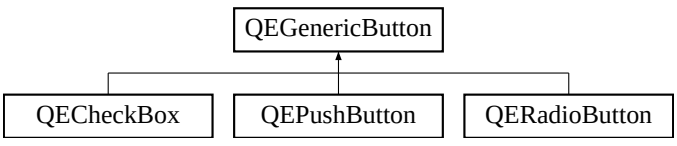
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.57 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 **setConfirmText** (QString confirmTextIn)
- QString **getConfirmText** ()
- 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 **setProgramStartupOption** (applicationLauncher::programStartupOptions programStartupOptionIn)
- applicationLauncher::programStartupOptions **getProgramStartupOption** ()
- void **setGuiName** (QString guiName)
- QString **getGuiName** ()
- void **setPrioritySubstitutions** (QString prioritySubstitutionsIn)
- QString **getPrioritySubstitutions** ()
- void **setCustomisationName** (QString customisationNameIn)
- QString **getCustomisationName** ()
- void **setCreationOption** (QEActionRequests::Options creationOption)
- QEActionRequests::Options **getCreationOption** ()
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()

Protected Member Functions

- void **connectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **setGenericButtonText** (const QString &text, QCaAlarmInfo &alarmInfo, QCaDateTime &, const unsigned int &variableIndex)
- void **userPressed** ()
- void **userReleased** ()
- void **userClicked** (bool checked)
- virtual updateOptions **getDefaultUpdateOption** ()=0
- void **startGui** (const QEActionRequests &request)
- void **setup** ()
- void **establishConnection** (unsigned int variableIndex)

Protected Attributes

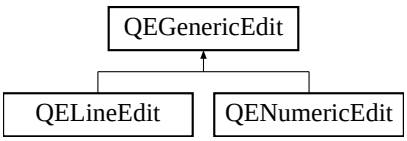
- applicationLauncher **programLauncher**

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

9.58 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

- void setVariableNameProperty (QString variableName)

  Access function for variable property - refer to variable property for details.
- QString getVariableNameProperty ()

  Access function for variable property - refer to variable property for details.
- void setVariableNameSubstitutionsProperty (QString variableNameSubstitutions)

  Access function for variableSubstitutions property - refer to variableSubstitutions property for details.
- QString getVariableNameSubstitutionsProperty ()

  Access function for variableSubstitutions property - refer to variableSubstitutions 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.
- 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)
- void **writeNow** ()

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

Protected Attributes

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

Properties

- QString **text**
- 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

9.58.1 Member Enumeration Documentation

9.58.1.1 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.

9.58.2 Constructor & Destructor Documentation

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

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

9.58.2.2 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.

9.58.3 Member Function Documentation

9.58.3.1 bool QEGenericEdit::getConfirmWrite ( )

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

9.58.3.2 bool QEGenericEdit::getSubscribe ( )

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

9.58.3.3 bool QEGenericEdit::getWriteOnEnter ( )

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

9.58.3.4 bool QEGenericEdit::getWriteOnFinish ( )

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

9.58.3.5 bool QEGenericEdit::getWriteOnLoseFocus ( )

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

9.58.3.6 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).

9.58.3.7 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)

9.58.3.8 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').

9.58.3.9 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).

9.58.3.10 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).

9.58.4 Property Documentation

9.58.4.1 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.

9.58.4.2 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).

9.58.4.3 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.

9.58.4.4 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.

9.58.4.5 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)

9.58.4.6 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'.

9.58.4.7 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.

9.58.4.8 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.

9.58.4.9 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.

9.58.4.10 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'.

9.58.4.11 QString QEGenericEdit::variable [read, write]

EPICS variable name (CA PV)

9.58.4.12 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.

9.58.4.13 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.

9.58.4.14 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.

9.58.4.15 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').

9.58.4.16 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).

9.58.4.17 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

9.59 QEGroupBox Class Reference

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)
- **QEGroupBox** (const QString &title, QWidget *parent=0)
- QSize **sizeHint** () const

Protected Member Functions

- virtual void **setSubstitutionsProperty** (QString macroSubstitutionsIn)
- QString **getSubstitutionsProperty** ()

Properties

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

9.59.1 Member Enumeration Documentation

9.59.1.1 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.

9.59.2 Property Documentation

9.59.2.1 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.

9.59.2.2 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.

9.59.2.3 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.

9.59.2.4 QString QEGroupBox::substitutedTitle [read, write]

Group box title text to be substituted. This text will be copied to the group box title text after applying any macro substitutions from the textSubstitutions property

9.59.2.5 QString QEGroupBox::textSubstitutions [read, write]

Text substitutions. These substitutions are applied to the 'substitutedTitle' property prior to copying it to the label text.

9.59.2.6 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'.

9.59.2.7 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.

9.59.2.8 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.

9.59.2.9 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.

9.59.2.10 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'.

9.59.2.11 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.

9.59.2.12 bool QEGroupBox::visible [read, write]

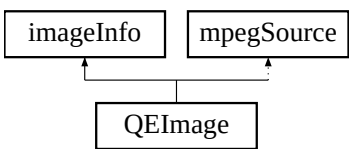
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.60 QEImage Class Reference

Inheritance diagram for QEImage:



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 **imageUses** { **IMAGE_USE_DISPLAY**, **IMAGE_USE_SAVE**, **IMAGE_USE_-DISPLAY_AND_SAVE** }
- enum resizeOptions { RESIZE_OPTION_ZOOM, RESIZE_OPTION_FIT }
- enum rotationOptions { ROTATION_0, ROTATION_90_RIGHT, ROTATION_90_-LEFT, ROTATION_180 }
- enum ellipseVariableDefinitions { BOUNDING_RECTANGLE, **CENTRE_AND_-SIZE** }
- enum UserLevels { User = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_-SCIENTIST, Engineer = userLevelTypes::USERLEVEL_ENGINEER }
- enum FormatOptions {
  Mono = imageDataFormats::MONO, Bayer = imageDataFormats::BAYERRG, Bay-erGB = imageDataFormats::BAYERGB, BayerBG = imageDataFormats::BAYERBG,
  BayerGR = imageDataFormats::BAYERGR, BayerRG = imageDataFormats::BAYERRG, rgb1 = imageDataFormats::RGB1, rgb2 = imageDataFormats::RGB2,
  rgb3 = imageDataFormats::RGB3, yuv444 = imageDataFormats::YUV444, yuv422 = imageDataFormats::YUV422, **yuv421** = imageDataFormats::YUV421 }
- enum EllipseVariableDefinitions { BoundingRectangle = BOUNDING_RECTANGLE, CenterAndSize = CENTRE_AND_SIZE }
- enum TargetOptions { DottedFullCrosshair = VideoWidget::CROSSHAIR1, SolidS-mallCrosshair = VideoWidget::CROSSHAIR2 }
- 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 }
- enum ProgramStartupOptionNames { None = applicationLauncher::PSO_NONE, Terminal = applicationLauncher::PSO_TERMINAL, LogOutput = applicationLauncher::PSO_-LOGOUTPUT, StdOutput = applicationLauncher::PSO_STDOUTPUT }

Public Slots

- void **setImageFile** (QString name)
- 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 targetClicked ()

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

- void imageDisplayPropsDestroyed (QObject ∗)

  Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

- void vSliceDisplayDestroyed (QObject ∗)

  Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

- void hSliceDisplayDestroyed (QObject ∗)

  Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

- void profileDisplayDestroyed (QObject ∗)

  Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

- void recorderDestroyed (QObject ∗)

  Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

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 **componentHostRequest** (const QEActionRequests &request)

Public Member Functions

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

  Destructor.
- selectOptions **getSelectionOption** ()
- void setBitDepth (unsigned int bitDepthIn)

  Access function for #bitDepth property - refer to #bitDepth property for details.
- unsigned int getBitDepth ()

  Access function for #bitDepth property - refer to #bitDepth property for details.
- void setFormatOption (imageDataFormats::formatOptions formatOption)

  Access function for formatOption property - refer to formatOption property for details.
- imageDataFormats::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 setUseFalseColour (bool pValue)

  Access function for useFalseColour property - refer to useFalseColour property for details.
- bool getUseFalseColour ()

  Access function for useFalseColour property - refer to useFalseColour 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 setEllipseMarkupColor (QColor markupColor)

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

- QColor getEllipseMarkupColor ()

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

- void setDisplayCursorPixelInfo (bool displayCursorPixelInfo)

  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 setLog (bool log)

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

- bool getLog ()

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

- void setEnableVertSliceSelection (bool enableVSliceSelection)

  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 enableHSliceSelection)

  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 setEnableArea1Selection (bool enableAreaSelectionIn)

  Access function for enableArea1Selection property - refer to enableArea1Selection property for details.

- bool getEnableArea1Selection ()

  Access function for enableArea1Selection property - refer to enableArea1Selection property for details.

- void setEnableArea2Selection (bool enableAreaSelectionIn)

  Access function for enableArea2Selection property - refer to enableArea2Selection property for details.

- bool getEnableArea2Selection ()

Access function for enableArea2Selection property - refer to enableArea2Selection property for details.

- void setEnableArea3Selection (bool enableAreaSelectionIn)

  Access function for enableArea3Selection property - refer to enableArea3Selection property for details.

- bool getEnableArea3Selection ()

  Access function for enableArea3Selection property - refer to enableArea3Selection property for details.

- void setEnableArea4Selection (bool enableAreaSelectionIn)

  Access function for enableArea4Selection property - refer to enableArea4Selection property for details.

- bool getEnableArea4Selection ()

  Access function for enableArea4Selection property - refer to enableArea4Selection 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 setEnableBeamSelection (bool enableBeamSelectionIn)

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

- bool getEnableBeamSelection ()

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

- void setEnableImageDisplayProperties (bool enableImageDisplayPropertiesIn)

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

- bool getEnableImageDisplayProperties ()

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

- void setEnableRecording (bool enableRecordingIn)

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

- bool getEnableRecording ()

  Access function for enableRecording property - refer to enableRecording 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.
- void setExternalControls (bool externalControlsIn)

  Access function for externalControls property - refer to externalControls property for details.
- bool getExternalControls ()

  Access function for externalControls property - refer to externalControls property for details.
- void setFullContextMenu (bool fullContextMenuIn)

  Access function for #fullContextMenu property - refer to #fullContextMenu property for details.
- bool getFullContextMenu ()

  Access function for #fullContextMenu property - refer to #fullContextMenu property for details.
- void setEnableProfilePresentation (bool enableProfilePresentationIn)

  Access function for #enableProfilePresentation property - refer to #enableProfilePresentation property for details.
- bool getEnableProfilePresentation ()

  Access function for #enableProfilePresentation property - refer to #enableProfilePresentation property for details.
- void setEnableHozSlicePresentation (bool enableHozSlicePresentationIn)

  Access function for #enableHozSlicePresentation property - refer to #enableHozSlicePresentation property for details.
- bool getEnableHozSlicePresentation ()

  Access function for #enableHozSlicePresentation property - refer to #enableHozSlicePresentation property for details.
- void setEnableVertSlicePresentation (bool enableVertSlicePresentationIn)

  Access function for #enableVertSlicePresentation property - refer to #enableVertSlicePresentation property for details.
- bool getEnableVertSlicePresentation ()

  Access function for #enableVertSlicePresentation property - refer to #enableVertSlicePresentation property for details.
- void setDisplayVertSliceSelection (bool displayVSliceSelection)

  Access function for displayVertSliceSelection property - refer to displayVertSliceSelection property for details.
- bool getDisplayVertSliceSelection ()

  Access function for displayVertSliceSelection property - refer to displayVertSliceSelection property for details.
- void setDisplayHozSliceSelection (bool displayHSliceSelection)

  Access function for displayHozSliceSelection property - refer to displayHozSliceSelection property for details.
- bool getDisplayHozSliceSelection ()

  Access function for displayHozSliceSelection property - refer to displayHozSliceSelection property for details.
- void setDisplayArea1Selection (bool displayAreaSelection)

Access function for displayArea1Selection property - refer to displayArea1Selection property for details.

- bool getDisplayArea1Selection ()

  Access function for displayArea1Selection property - refer to displayArea1Selection property for details.

- void setDisplayArea2Selection (bool displayAreaSelection)

  Access function for displayArea2Selection property - refer to displayArea2Selection property for details.

- bool getDisplayArea2Selection ()

  Access function for displayArea2Selection property - refer to displayArea2Selection property for details.

- void setDisplayArea3Selection (bool displayAreaSelection)

  Access function for displayArea3Selection property - refer to displayArea3Selection property for details.

- bool getDisplayArea3Selection ()

  Access function for displayArea3Selection property - refer to displayArea3Selection property for details.

- void setDisplayArea4Selection (bool displayAreaSelection)

  Access function for displayArea4Selection property - refer to displayArea4Selection property for details.

- bool getDisplayArea4Selection ()

  Access function for displayArea4Selection property - refer to displayArea4Selection property for details.

- void setDisplayProfileSelection (bool displayProfileSelection)

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

- bool getDisplayProfileSelection ()

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

- void setDisplayTargetSelection (bool displayTargetSelection)

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

- bool getDisplayTargetSelection ()

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

- void setDisplayBeamSelection (bool displayBeamSelection)

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

- bool getDisplayBeamSelection ()

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

- void setDisplayEllipse (bool displayEllipse)

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

- bool getDisplayEllipse ()

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

- ellipseVariableDefinitions getEllipseVariableDefinition ()

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

- void setEllipseVariableDefinition (ellipseVariableDefinitions def)

  Access function for ellipseVariableDefinition property - refer to ellipseVariableDefinition property for details.
- void setDisplayMarkups (bool displayMarkupsIn)

  Access function for #displayMarkups property - refer to #displayMarkups property for details.
- bool getDisplayMarkups ()

  Access function for #displayMarkups property - refer to #displayMarkups property for details.
- void setProgram1 (QString program)

  Access function for program1 property - refer to program1 property for details.
- QString getProgram1 ()

  Access function for program1 property - refer to program1 property for details.
- void setProgram2 (QString program)

  Access function for program2 property - refer to program2 property for details.
- QString getProgram2 ()

  Access function for program2 property - refer to program2 property for details.
- void setArguments1 (QStringList arguments)

  Access function for arguments1 property - refer to arguments1 property for details.
- QStringList getArguments1 ()

  Access function for arguments1 property - refer to arguments1 property for details.
- void setArguments2 (QStringList arguments)

  Access function for arguments2 property - refer to arguments2 property for details.
- QStringList getArguments2 ()

  Access function for arguments2 property - refer to arguments2 property for details.
- void setProgramStartupOption1 (applicationLauncher::programStartupOptions programStartupOption)

  Access function for programStartupOption1 property - refer to programStartupOption1 property for details.
- applicationLauncher::programStartupOptions getProgramStartupOption1 ()

  Access function for programStartupOption1 property - refer to programStartupOption1 property for details.
- void setProgramStartupOption2 (applicationLauncher::programStartupOptions programStartupOption)

  Access function for programStartupOption2 property - refer to programStartupOption2 property for details.
- applicationLauncher::programStartupOptions getProgramStartupOption2 ()

  Access function for programStartupOption2 property - refer to programStartupOption2 property for details.
- QString getHozSliceLegend ()

  Access function for hozSliceLegend property - refer to hozSliceLegend property for details.
- void setHozSliceLegend (QString legend)

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

- QString getVertSliceLegend ()

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

- void setVertSliceLegend (QString legend)

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

- QString getprofileLegend ()

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

- void setProfileLegend (QString legend)

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

- QString getAreaSelection1Legend ()

  Access function for areaSelection1Legend property - refer to areaSelection1Legend property for details.

- void setAreaSelection1Legend (QString legend)

  Access function for areaSelection1Legend property - refer to areaSelection1Legend property for details.

- QString getAreaSelection2Legend ()

  Access function for areaSelection2Legend property - refer to areaSelection2Legend property for details.

- void setAreaSelection2Legend (QString legend)

  Access function for areaSelection2Legend property - refer to areaSelection2Legend property for details.

- QString getAreaSelection3Legend ()

  Access function for areaSelection3Legend property - refer to areaSelection3Legend property for details.

- void setAreaSelection3Legend (QString legend)

  Access function for areaSelection3Legend property - refer to areaSelection3Legend property for details.

- QString getAreaSelection4Legend ()

  Access function for areaSelection4Legend property - refer to areaSelection4Legend property for details.

- void setAreaSelection4Legend (QString legend)

  Access function for areaSelection4Legend property - refer to areaSelection4Legend property for details.

- QString getTargetLegend ()

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

- void setTargetLegend (QString legend)

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

- QString getBeamLegend ()

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

- void setBeamLegend (QString legend)

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

- QString getEllipseLegend ()

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

- void setEllipseLegend (QString legend)

  Access function for ellipseLegend property - refer to ellipseLegend property for details.
- bool getFullScreen ()

  Access function for #fullScreen property - refer to #fullScreen property for details.
- void setFullScreen (bool fullScreenIn)

  Access function for #fullScreen property - refer to #fullScreen property for details.
- void setSubstitutedUrl (QString urlIn)

  Access function for URL property - refer to URL property for deta.
- QString getSubstitutedUrl ()

  Access function for URL property - refer to URL property for deta.
- 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 setBitDepthProperty (unsigned int bitDepth)

  Access function for #bitDepth property - refer to #bitDepth property for details.
- unsigned int getBitDepthProperty ()

  Access function for #bitDepth property - refer to #bitDepth property for details.
- EllipseVariableDefinitions getEllipseVariableDefinitionProperty ()

  Access function for #EllipseVariableDefinition property - refer to #EllipseVariableDefinition property for details.
- void setEllipseVariableDefinitionProperty (EllipseVariableDefinitions variableUsage)

  Access function for EllipseVariableDefinitions property - refer to EllipseVariableDefinitions property for details.
- TargetOptions getTargetOptionProperty ()

  Access function for targetOption property - refer to targetOption property for details.
- void setTargetOptionProperty (TargetOptions option)

  Access function for targetOption property - refer to targetOption property for details.
- TargetOptions getBeamOptionProperty ()

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

- void setBeamOptionProperty (TargetOptions option)

  Access function for beamOption property - refer to beamOption 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.
- void setProgramStartupOptionProperty1 (ProgramStartupOptionNames programStartupOption)

  Access function for #ProgramStartupOptionNames1 property - refer to #ProgramStartupOptionNames1 property for details.
- ProgramStartupOptionNames getProgramStartupOptionProperty1 ()

  Access function for #ProgramStartupOptionNames1 property - refer to #ProgramStartupOptionNames1 property for details.
- void setProgramStartupOptionProperty2 (ProgramStartupOptionNames programStartupOption)

  Access function for #ProgramStartupOptionNames2 property - refer to #ProgramStartupOptionNames2 property for details.
- ProgramStartupOptionNames getProgramStartupOptionProperty2 ()

  Access function for #ProgramStartupOptionNames2 property - refer to #ProgramStartupOptionNames2 property for details.

Protected Types

- enum **variableIndexes** {

  IMAGE_VARIABLE, **FORMAT_VARIABLE**, **BIT_DEPTH_VARIABLE**, **WIDTH_VARIABLE**,

  HEIGHT_VARIABLE, **NUM_DIMENSIONS_VARIABLE**, **DIMENSION_0_VARIABLE**, **DIMENSION_1_VARIABLE**,

  DIMENSION_2_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, **PROFILE_H_VARIABLE**, **PROFILE_H_THICKNESS_VARIABLE**, **PROFILE_V_VARIABLE**,

PROFILE_V_THICKNESS_VARIABLE, **LINE_PROFILE_X1_VARIABLE**, **LINE_-PROFILE_Y1_VARIABLE**, **LINE_PROFILE_X2_VARIABLE**,

LINE_PROFILE_Y2_VARIABLE, **LINE_PROFILE_THICKNESS_VARIABLE**, **PROFILE_-H_ARRAY**, **PROFILE_V_ARRAY**,

PROFILE_LINE_ARRAY, **ELLIPSE_X_VARIABLE**, **ELLIPSE_Y_VARIABLE**, **ELLIPSE_-W_VARIABLE**,

ELLIPSE_H_VARIABLE, **QEIMAGE_NUM_VARIABLES** }

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- QImage **copyImage** ()
- void **redisplayAllMarkups** ()
- 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

- QEStringFormatting **stringFormatting**
- QEIntegerFormatting **integerFormatting**
- QEFloatingFormatting **floatingFormatting**
- resizeOptions **resizeOption**
- int zoom

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

Properties

- QString imageVariable
- QString formatVariable
- QString bitDepthVariable
- QString widthVariable
- QString heightVariable

- QString dimensionsVariable
- QString dimension1Variable
- QString dimension2Variable
- QString dimension3Variable
- 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 profileHozVariable
- QString profileHozThicknessVariable
- QString profileVertVariable
- QString profileVertThicknessVariable
- QString lineProfileX1Variable
- QString lineProfileY1Variable
- QString lineProfileX2Variable
- QString lineProfileY2Variable
- QString lineProfileThicknessVariable
- QString profileHozArrayVariable
- QString profileVertArrayVariable
- QString lineProfileArrayVariable
- QString ellipseXVariable
- QString ellipseYVariable
- QString ellipseWVariable
- QString ellipseHVariable
- 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 enableProfileSelection
- bool enableArea1Selection
- bool enableArea2Selection
- bool enableArea3Selection
- bool enableArea4Selection
- bool enableTargetSelection
- bool enableBeamSelection
- QString hozSliceLegend

  Name of horizontal slice profile markup.
- QString vertSliceLegend

  Name of vertical slice profile markup.
- QString profileLegend

  Name of arbitrary priofile markup.
- QString areaSelection1Legend

  Name of area selection 1 markup.
- QString areaSelection2Legend

  Name of area selection 2 markup.
- QString areaSelection3Legend

  Name of area selection 3 markup.
- QString areaSelection4Legend

  Name of area selection 4 markup.
- QString targetLegend

  Name of target markup.
- QString beamLegend

  Name of beam markup.
- QString ellipseLegend

  Name of ellipse markup.
- bool displayVertSliceSelection
- bool displayHozSliceSelection
- bool displayProfileSelection
- bool displayArea1Selection
- bool displayArea2Selection
- bool displayArea3Selection

- bool displayArea4Selection
- bool displayTargetSelection
- bool displayBeamSelection
- bool displayEllipse
- EllipseVariableDefinitions ellipseVariableDefinition

  Definition of how ellipse variables are to be used.
- TargetOptions targetOption

  Definition of target markup options.
- TargetOptions beamOption

  Definition of beam markup options.
- bool displayCursorPixelInfo
- bool contrastReversal
- bool logBrightness
- bool showTime
- bool useFalseColour
- QColor vertSliceColor
- QColor hozSliceColor
- QColor profileColor
- QColor areaColor
- QColor beamColor
- QColor targetColor
- QColor timeColor
- QColor ellipseColor
- ResizeOptions resizeOption
- RotationOptions rotation
- bool verticalFlip
- bool horizontalFlip
- int initialHosScrollPos
- bool enableImageDisplayProperties

  If true, the local Image Display Properties controls are displayed.
- bool enableRecording

  If true, the recording controls are displayed.
- bool autoBrightnessContrast
- bool externalControls
- bool briefInfoArea
- QString program1
- QStringList arguments1
- ProgramStartupOptionNames programStartupOption1
- QString program2
- QStringList arguments2
- ProgramStartupOptionNames programStartupOption2
- QString URL

9.60.1 Member Enumeration Documentation

9.60.1.1 enum QEImage::ellipseVariableDefinitions

Options for the use of ellipse markup variables.

Enumerator:

BOUNDING_RECTANGLE Variables define bounding rectagle of ellipse.

9.60.1.2 enum QEImage::EllipseVariableDefinitions

User friendly enumerations for ellipseVariableDefinition property - refer to ellipseVariableDefinition property for details.

Enumerator:

BoundingRectangle Refer to BOUNDING_RECTANGLE for details.

CenterAndSize Refer to CENTRE_AND_SIZE for details.

9.60.1.3 enum QEImage::FormatOptions

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

Enumerator:

Mono Grey scale.

Bayer Colour (Bayer Red Green)

BayerGB Colour (Bayer Green Blue)

BayerBG Colour (Bayer Blue Green)

BayerGR Colour (Bayer Green Red)

BayerRG Colour (Bayer Red Green)

rgb1 Colour (24 bit RGB)

rgb2 Colour (??? bit RGB)

rgb3 Colour (??? bit RGB)

yuv444 Colour (???)

yuv422 Colour (???)

9.60.1.4 enum QEImage::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a termainal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send doutput to standard output and standard error.

9.60.1.5 enum QEImage::ResizeOptions

User friendly enumerations for #resizeOption property

Enumerator:

Zoom Zoom to selected percentage.

Fit Zoom to fit the current window size.

9.60.1.6 enum QEImage::resizeOptions

Image resize options

Enumerator:

RESIZE_OPTION_ZOOM Zoom to selected percentage.

RESIZE_OPTION_FIT Zoom to fit the current window size.

9.60.1.7 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.

9.60.1.8 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.

9.60.1.9 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.

9.60.1.10 enum QEImage::TargetOptions

User friendly enumerations for #targetOptions property - refer to #targetOptions property for details.

Enumerator:

DottedFullCrosshair Refer to CROSSHAIR1 for details.

SolidSmallCrosshair Refer to CROSSHAIR2 for details.

9.60.1.11 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.

9.60.2 Constructor & Destructor Documentation

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

Create without a variable. Use setVariableName'n'Property() - where 'n' is a number from 0 to 40 - 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.

9.60.2.2 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.

9.60.3 Member Function Documentation

9.60.3.1 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.

9.60.4 Member Data Documentation

9.60.4.1 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.

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

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

9.60.5 Property Documentation

9.60.5.1 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.

9.60.5.2 QColor QEImage::areaColor [read, write]

Used to select the color of the area selection markups.

9.60.5.3 QStringList QEImage::arguments1 [read, write]

Arguments for program specified in the 'program1' property.

9.60.5.4 QStringList QEImage::arguments2 [read, write]

Arguments for program specified in the 'program2' property.

9.60.5.5 bool QEImage::autoBrightnessContrast [read, write]

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.

9.60.5.6 QColor QEImage::beamColor [read, write]

Used to select the color of the beam marker.

9.60.5.7 QString QEImage::beamXVariable [read, write]

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

9.60.5.8 QString QEImage::beamYVariable [read, write]

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

9.60.5.9 QString QEImage::bitDepthVariable [read, write]

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

9.60.5.10 bool QEImage::briefInfoArea [read, write]

If true, the information area willl be brief (one row)

9.60.5.11 QString QEImage::clippingHighVariable [read, write]

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

9.60.5.12 QString QEImage::clippingLowVariable [read, write]

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

9.60.5.13 QString QEImage::clippingOnOffVariable [read, write]

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

9.60.5.14 bool QEImage::contrastReversal [read, write]

If true, the image will undergo contrast reversal.

9.60.5.15 QString QEImage::dimension1Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the first area detector dimension of the image. If there are 2 dimensions, this will be the image width. If there are 3 dimensions, this will be the number of elements per pixel.

9.60.5.16 QString QEImage::dimension2Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the second area detector dimension of the image. If there are 2 dimensions, this will be the image height. If there are 3 dimensions, this will be the image width.

9.60.5.17 QString QEImage::dimension3Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the third area detector dimension of the image. If there are 3 dimensions, this will be the image height.

9.60.5.18 QString QEImage::dimensionsVariable [read, write]

EPICS variable name (CA PV). This variable is used to read the number of area detector dimensions of the image. If used, this will be 2 (one element per pixel arranged by width and height) or 3 (multiple elements per pixel arranged by pixel, width and height)

9.60.5.19 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.

9.60.5.20 bool QEImage::displayArea1Selection [read, write]

If true, selected area 1 will be displayed on the image. Note, this property is ignored unless the enableArea1Selection property is true.

9.60.5.21 bool QEImage::displayArea2Selection [read, write]

If true, selected area 2 will be displayed on the image. Note, this property is ignored unless the enableArea2Selection property is true.

9.60.5.22 bool QEImage::displayArea3Selection [read, write]

If true, selected area 3 will be displayed on the image. Note, this property is ignored unless the enableArea3Selection property is true.

9.60.5.23 bool QEImage::displayArea4Selection [read, write]

If true, selected area 4 will be displayed on the image. Note, this property is ignored unless the enableArea4Selection property is true.

9.60.5.24 bool QEImage::displayBeamSelection [read, write]

If true, beam selection will be displayed on the image. Note, this property is ignored unless the enableBeamSelection property is true.

9.60.5.25 bool QEImage::displayCursorPixelInfo [read, write]

If true, an area will be presented under the image with textual information about the pixel under the cursor, and for other selections such as selected areas.

9.60.5.26 bool QEImage::displayEllipse [read, write]

If true, the ellipse markup will be displayed on the image.

9.60.5.27 bool QEImage::displayHozSliceSelection [read, write]

If true, the selected horizontal slice will be displayed on the image. Note, this property is ignored unless the enableHozSliceSelection property is true.

9.60.5.28 bool QEImage::displayProfileSelection [read, write]

If true, the selected arbirtary line will be displayed on the image. Note, this property is ignored unless the enableProfileSelection property is true.

9.60.5.29 bool QEImage::displayTargetSelection [read, write]

If true, target selection will be displayed on the image. Note, this property is ignored unless the enableTargetSelection property is true.

9.60.5.30 bool QEImage::displayVertSliceSelection [read, write]

If true, the selected vertical slice will be displayed on the image. Note, this property is ignored unless the enableVertSliceSelection property is true.

9.60.5.31 QColor QEImage::ellipseColor [read, write]

Used to select the color of the ellipse marker.

9.60.5.32 QString QEImage::ellipseHVariable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse height

9.60.5.33 QString QEImage::ellipseWVariable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse width.

9.60.5.34 QString QEImage::ellipseXVariable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse X (center or top left corner of bounding rectangle depending on property ellipseDefinition).

9.60.5.35 QString QEImage::ellipseYVariable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse Y (center or top left corner of bounding rectangle depending on property ellipseDefinition).

9.60.5.36 bool QEImage::enableArea1Selection [read, write]

If true, the user will be able to select area 1. These are used for selection of Region of Interests, and for zooming to area 1

9.60.5.37 bool QEImage::enableArea2Selection [read, write]

If true, the user will be able to select area 2. These are used for selection of Region of Interests, and for zooming to area 2

9.60.5.38 bool QEImage::enableArea3Selection [read, write]

If true, the user will be able to select area 3. These are used for selection of Region of Interests, and for zooming to area 3

9.60.5.39 bool QEImage::enableArea4Selection [read, write]

If true, the user will be able to select area 4. These are used for selection of Region of Interests, and for zooming to area 4

9.60.5.40 bool QEImage::enableBeamSelection [read, write]

If true, the user will be able to select points on the image to mark a beam position. This can be used for automatic beam positioning.

9.60.5.41 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, and write the position of the slice to the optional variable specified by the profileHozVariable property. The profile will only be presented to the user if #enableHozSlicePresentation property is true.

9.60.5.42 bool QEImage::enableProfileSelection [read, write]

If true, the option to select an arbitrary line through any part of the image will be available to the user. This will be used to generate a pixel profile.

9.60.5.43 bool QEImage::enableTargetSelection [read, write]

If true, the user will be able to select points on the image to mark a target position. This can be used for automatic beam positioning.

9.60.5.44 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 horizontal pixel profile, and write the position of the slice to the optional variable specified by the profileVertVariable property. The profile will only be presented to the user if #enableVertSlicePresentation property is true.

9.60.5.45 bool QEImage::externalControls [read, write]

If true, image controls and views such as brightness controls and profile plots are hosted by the application as dock windows, toolbars, etc. Refer to the #ContainerProfile class and the #windowCustomisation class to see how this class asks an application to act as a host.

9.60.5.46 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.)

9.60.5.47 QString QEImage::formatVariable [read, write]

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

9.60.5.48 QString QEImage::heightVariable [read, write]

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

9.60.5.49 bool QEImage::horizontalFlip [read, write]

If true, flip image horizontally.

9.60.5.50 QColor QEImage::hozSliceColor [read, write]

Used to select the color of the horizontal slice markup.

9.60.5.51 QString QEImage::imageVariable [read, write]

EPICS variable name (CA PV). This variable is used as the source the image waveform.

9.60.5.52 int QEImage::initialHosScrollPos [read, write]

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

9.60.5.53 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.

Bit depth. Note, EPICS data type size will typically be adequate for the number of bits required (one byte for up to 8 bits, 2 bytes for up to 16 bits, etc), but can be larger (for example, 4 bytes for 24 bits) and may be larger than nessesary (4 bytes for 8 bits).

9.60.5.54 QString QEImage::lineProfileArrayVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile array.

9.60.5.55 QString QEImage::lineProfileThicknessVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end Y.

9.60.5.56 QString QEImage::lineProfileX1Variable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile start X.

9.60.5.57 QString QEImage::lineProfileX2Variable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end X.

9.60.5.58 QString QEImage::lineProfileY1Variable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile start Y.

9.60.5.59 QString QEImage::lineProfileY2Variable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end Y.

9.60.5.60 bool QEImage::logBrightness [read, write]

If true, the image will be displayed using a logarithmic brightness scale.

9.60.5.61 QColor QEImage::profileColor [read, write]

Used to select the color of the arbitrarty profile line markup.

9.60.5.62 QString QEImage::profileHozArrayVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile array.

9.60.5.63 QString QEImage::profileHozThicknessVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile thickness.

9.60.5.64 QString QEImage::profileHozVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile.

9.60.5.65 QString QEImage::profileVertArrayVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile array.

9.60.5.66 QString QEImage::profileVertThicknessVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile.

9.60.5.67 QString QEImage::profileVertVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile.

9.60.5.68 QString QEImage::program1 [read, write]

Program to run when a request is made to pass on the current image to the first external application. No attempt to run a program is made if this property is empty. Example: paint.exe

9.60.5.69 QString QEImage::program2 [read, write]

Program to run when a request is made to pass on the current image to the second external application. No attempt to run a program is made if this property is empty. Example: paint.exe

9.60.5.70 ProgramStartupOptionNames QEImage::programStartupOption1 [read, write]

Startup options for the program specified in the 'program1' property. Just run the command, run the command within a terminal, or display the output in QE message system.

9.60.5.71 ProgramStartupOptionNames QEImage::programStartupOption2 [read, write]

Startup options for the program specified in the 'program2' property. Just run the command, run the command within a terminal, or display the output in QE message system.

9.60.5.72 QString QEImage::regionOfInterest1HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest height.

9.60.5.73 QString QEImage::regionOfInterest1WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest width.

9.60.5.74 QString QEImage::regionOfInterest1XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest X position.

9.60.5.75 QString QEImage::regionOfInterest1YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the first region of interest Y position.

9.60.5.76 QString QEImage::regionOfInterest2HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest height.

9.60.5.77 QString QEImage::regionOfInterest2WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest width.

9.60.5.78 QString QEImage::regionOfInterest2XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest X position.

9.60.5.79 QString QEImage::regionOfInterest2YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the second region of interest Y position.

9.60.5.80 QString QEImage::regionOfInterest3HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest height.

9.60.5.81 QString QEImage::regionOfInterest3WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest width.

9.60.5.82 QString QEImage::regionOfInterest3XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest X position.

9.60.5.83 QString QEImage::regionOfInterest3YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the third region of interest Y position.

9.60.5.84 QString QEImage::regionOfInterest4HVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest height.

9.60.5.85 QString QEImage::regionOfInterest4WVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest width.

9.60.5.86 QString QEImage::regionOfInterest4XVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest X position.

9.60.5.87 QString QEImage::regionOfInterest4YVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the fourth region of interest Y position.

9.60.5.88 ResizeOptions QEImage::resizeOption [read, write]

Resize option. Zoom to zoom to the percentage given by the zoom property, or fit to the window size.

9.60.5.89 RotationOptions QEImage::rotation [read, write]

Image rotation option.

9.60.5.90 bool QEImage::showTime [read, write]

If true, the image timestamp will be written in the top left of the image.

9.60.5.91 QColor QEImage::targetColor [read, write]

Used to select the color of the target marker.

9.60.5.92 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.

9.60.5.93 QString QEImage::targetXVariable [read, write]

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

9.60.5.94 QString QEImage::targetYVariable [read, write]

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

9.60.5.95 QColor QEImage::timeColor [read, write]

Used to select the color of the timestamp.

9.60.5.96 QString QEImage::URL [read, write]

MPEG stream URL. If this is specified, this will be used as the source of the image in preference to variables (variables defining the image data, width, and height will be ignored)

9.60.5.97 bool QEImage::useFalseColour [read, write]

If true, the apply false colour to the image.

9.60.5.98 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'.

9.60.5.99 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.

9.60.5.100 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.

9.60.5.101 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.

9.60.5.102 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'.

9.60.5.103 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.

9.60.5.104 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.

9.60.5.105 bool QEImage::verticalFlip [read, write]

If true, flip image vertically.

9.60.5.106 QColor QEImage::vertSliceColor [read, write]

Used to select the color of the vertical slice markup.

9.60.5.107 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.

9.60.5.108 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

9.61 QEImageMarkupThickness Class Reference

Public Member Functions

- **QEImageMarkupThickness** (QWidget ∗parent=0)
- void **setThickness** (unsigned int thicknessIn)
- unsigned int **getThickness** ()

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

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

9.62 QEImageOptionsDialog Class Reference

Signals

- void **optionChange** (imageContextMenu::imageContextMenuOptions option, bool checked)

Public Member Functions

- **QEImageOptionsDialog** (QWidget ∗parent=0)
- void **initialise** ()
- void **optionSet** (imageContextMenu::imageContextMenuOptions option, bool checked)
- bool **optionGet** (imageContextMenu::imageContextMenuOptions option)

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

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

9.63 QELabel Class Reference

`#include <QELabel.h>`

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, 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 = 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
- QString **text**
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7

9.63.1 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.

9.63.2 Member Enumeration Documentation

9.63.2.1 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.

9.63.2.2 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.

9.63.2.3 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.

9.63.2.4 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.

9.63.2.5 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.

9.63.2.6 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.

9.63.3 Constructor & Destructor Documentation

9.63.3.1 QELabel::QELabel (QWidget * *parent* = 0)

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

9.63.3.2 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.

9.63.4 Member Function Documentation

9.63.4.1 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.

9.63.5 Property Documentation

9.63.5.1 bool QELabel::addUnits [read, write]

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

9.63.5.2 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.

9.63.5.3 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'.

9.63.5.4 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.

9.63.5.5 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).

9.63.5.6 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.

9.63.5.7 bool QELabel::leadingZero [read, write]

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

9.63.5.8 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"

9.63.5.9 Notations QELabel::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.63.5.10 QPixmap QELabel::pixmap0 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 0.

9.63.5.11 QPixmap QELabel::pixmap1 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 1.

9.63.5.12 QPixmap QELabel::pixmap2 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 2.

9.63.5.13 QPixmap QELabel::pixmap3 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 3.

9.63.5.14 QPixmap QELabel::pixmap4 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 4.

9.63.5.15 QPixmap QELabel::pixmap5 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 5.

9.63.5.16 QPixmap QELabel::pixmap6 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 6.

9.63.5.17 QPixmap QELabel::pixmap7 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 7.

9.63.5.18 int QELabel::precision [read, write]

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

9.63.5.19 bool QELabel::trailingZeros [read, write]

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

9.63.5.20 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.

9.63.5.21 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.

9.63.5.22 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'.

9.63.5.23 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.

9.63.5.24 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.

9.63.5.25 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.

9.63.5.26 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'.

9.63.5.27 QString QELabel::variable [read, write]

EPICS variable name (CA PV)

9.63.5.28 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.

9.63.5.29 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.

9.63.5.30 bool QELabel::visible [read, write]

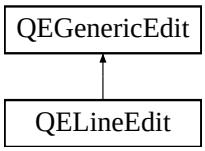
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.64 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

9.64.1 Member Enumeration Documentation

9.64.1.1 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.

9.64.1.2 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.

9.64.1.3 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.

9.64.2 Constructor & Destructor Documentation

9.64.2.1 QELineEdit::QELineEdit (QWidget * *parent* = 0)

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

9.64.2.2 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.

9.64.3 Member Function Documentation

9.64.3.1 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.

9.64.4 Property Documentation

9.64.4.1 bool QELineEdit::addUnits [read, write]

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

9.64.4.2 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'.

9.64.4.3 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).

9.64.4.4 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.

9.64.4.5 bool QELineEdit::leadingZero [read, write]

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

9.64.4.6 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"

9.64.4.7 Notations QELineEdit::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.64.4.8 int QELineEdit::precision [read, write]

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

9.64.4.9 bool QELineEdit::trailingZeros [read, write]

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

9.64.4.10 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

9.65 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

9.66 QELink Class Reference

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 long &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 **setConditionProperty** (ConditionNames condition)
- ConditionNames **getConditionProperty** ()

Protected Attributes

- conditions **condition**
- QVariant **comparisonValue**
- bool **signalTrue**
- bool **signalFalse**
- QVariant **outTrueValue**
- QVariant **outFalseValue**

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

9.67 QELog Class Reference

Public Types

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

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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- 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 **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

- 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 Attributes

- _QTableWidgetLog * **qTableWidgetLog**
- QCheckBox * **qCheckBoxInfoMessage**
- QCheckBox * **qCheckBoxWarningMessage**
- QCheckBox * **qCheckBoxErrorMessage**
- QPushButton * **qPushButtonClear**
- QPushButton * **qPushButtonSave**
- QColor **qColorInfo**
- QColor **qColorWarning**
- QColor **qColorError**
- bool **scrollToBottom**
- int **optionsLayout**

Properties

- bool **showColumnTime**
- bool **showColumnType**
- bool **showColumnMessage**
- bool **showMessageFilter**
- bool **showClear**
- bool **showSave**
- optionsLayoutProperty **optionsLayout**
- QColor **infoColor**
- QColor **warningColor**
- QColor **errorColor**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**
- unsigned int
- bool variableAsToolTip
- bool allowDrop
- bool visible

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

9.67.1 Member Enumeration Documentation

9.67.1.1 enum QELog::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.

9.67.2 Property Documentation

9.67.2.1 bool QELog::allowDrop [read, write]

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

9.67.2.2 bool QELog::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.

9.67.2.3 unsigned QELog::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.

9.67.2.4 UserLevels QELog::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'.

9.67.2.5 QString QELog::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.

9.67.2.6 QString QELog::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.

9.67.2.7 QString QELog::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.

9.67.2.8 UserLevels QELog::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'.

9.67.2.9 bool QELog::variableAsToolTip [read, write]

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

9.67.2.10 bool QELog::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/QELog/QELog.h
- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.cpp

9.68 QELogin Class Reference

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

9.69 QELoginDialog Class Reference

Public Member Functions

- **QELoginDialog** (QELogin *ownerIn)

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

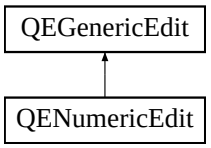
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.70 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:



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** ()
- 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 QEFixedPointRadix::Radicies value)
- QEFixedPointRadix::Radicies **getRadix** ()
- void **setSeparator** (const QEFixedPointRadix::Separators value)
- QEFixedPointRadix::Separators **getSeparator** ()

Protected Member Functions

- 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
- QEFixedPointRadix::Radicies radix

  Specify radix, default is Decimal.
- QEFixedPointRadix::Separators separator

  Specify digit 'thousands' separator character, default is none.
- int precision
- int leadingZeros
- double minimum
- double maximum
- bool addUnits

Friends

- class **NumericValidator**

9.70.1 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.

9.70.2 Constructor & Destructor Documentation

9.70.2.1 QENumericEdit::QENumericEdit (QWidget * *parent* = 0)

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

9.70.2.2 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.

9.70.3 Member Function Documentation

9.70.3.1 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.

9.70.4 Property Documentation

9.70.4.1 bool QENumericEdit::addUnits [read, write]

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

9.70.4.2 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.

9.70.4.3 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.

9.70.4.4 double QENumericEdit::maximum [read, write]

Speficies the maximum allowed value. This is only used if autoScale is false.

9.70.4.5 double QENumericEdit::minimum [read, write]

Speficies the mimimum allowed value. This is only used if autoScale is false.

9.70.4.6 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

9.71 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

9.72 QEPeriodic Class Reference

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

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**

9.72.1 Member Enumeration Documentation

9.72.1.1 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.

9.72.2 Member Function Documentation

9.72.2.1 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.

9.72.2.2 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.

9.72.3 Member Data Documentation

9.72.3.1 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.

9.72.4 Property Documentation

9.72.4.1 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.

9.72.4.2 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.

9.72.4.3 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.

9.72.4.4 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.

9.72.4.5 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)

9.72.4.6 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'.

9.72.4.7 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.

9.72.4.8 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.

9.72.4.9 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.

9.72.4.10 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'.

9.72.4.11 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.

9.72.4.12 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.

9.72.4.13 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.

9.72.4.14 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.

9.72.4.15 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

9.73 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

9.74 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

9.75 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

9.76 QEPlot Class Reference

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)
- QSize **sizeHint** () const
- 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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

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**

9.76.1 Member Enumeration Documentation

9.76.1.1 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.

9.76.2 Member Function Documentation

9.76.2.1 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.

9.76.2.2 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.

9.76.3 Member Data Documentation

9.76.3.1 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.

9.76.4 Property Documentation

9.76.4.1 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.

9.76.4.2 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.

9.76.4.3 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'.

9.76.4.4 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.

9.76.4.5 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.

9.76.4.6 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.

9.76.4.7 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'.

9.76.4.8 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.

9.76.4.9 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.

9.76.4.10 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.

9.76.4.11 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.

9.76.4.12 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.

9.76.4.13 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.

9.76.4.14 bool QEPlot::visible [read, write]

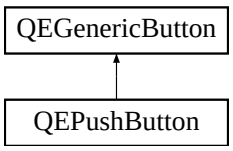
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.77 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, 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 }
- 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 ProgramStartupOptionNames { None = applicationLauncher::PSO_NONE, Terminal = applicationLauncher::PSO_TERMINAL, LogOutput = applicationLauncher::PSO_-LOGOUTPUT, StdOutput = applicationLauncher::PSO_STDOUTPUT }
- enum CreationOptionNames {

  Open = QEActionRequests::OptionOpen, NewTab = QEActionRequests::OptionNewTab, NewWindow = QEActionRequests::OptionNewWindow, DockTop = QEActionRequests::OptionTopDockWindow,

  DockBottom = QEActionRequests::OptionBottomDockWindow, DockLeft = QE-ActionRequests::OptionLeftDockWindow, DockRight = QEActionRequests::OptionRightDockWindow, DockTopTabbed = QEActionRequests::OptionTopDockWindowTabbed,

  DockBottomTabbed = QEActionRequests::OptionBottomDockWindowTabbed, Dock-LeftTabbed = QEActionRequests::OptionLeftDockWindowTabbed, DockRightTabbed = QEActionRequests::OptionRightDockWindowTabbed, DockFloating = QEAction-Requests::OptionFloatingDockWindow }

  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 requestAction (const QEActionRequests &request)

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 (const QEActionRequests &request)

  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)
- void programCompleted ()

  Program started by button has compelted.

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

9.77.1 Member Enumeration Documentation

9.77.1.1 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.

9.77.1.2 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.

DockTop Open new GUI in a top dock window.

DockBottom Open new GUI in a bottom dock window.

DockLeft Open new GUI in a left dock window.

DockRight Open new GUI in a right dock window.

DockTopTabbed Open new GUI in a top dock window (tabbed with any existing dock in that area)

DockBottomTabbed Open new GUI in a bottom dock window (tabbed with any existing dock in that area)

DockLeftTabbed Open new GUI in a left dock window (tabbed with any existing dock in that area)

DockRightTabbed Open new GUI in a right dock window (tabbed with any existing dock in that area)

DockFloating Open new GUI in a floating dock window.

9.77.1.3 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.

9.77.1.4 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.

9.77.1.5 enum QEPushButton::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a termainal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send doutput to standard output and standard error.

9.77.1.6 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)

9.77.1.7 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.

9.77.2 Constructor & Destructor Documentation

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

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

9.77.2.2 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.

9.77.3 Member Function Documentation

9.77.3.1 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)

9.77.3.2 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.

9.77.3.3 void QEPushButton::pressed (int *value*) [signal]

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

9.77.3.4 void QEPushButton::released (int *value*) [signal]

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

9.77.3.5 void QEPushButton::requestAction (const QEActionRequests & *request*) [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.

9.77.4 Property Documentation

9.77.4.1 bool QEPushButton::addUnits [read, write]

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

9.77.4.2 Qt::Alignment QEPushButton::alignment [read, write]

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

9.77.4.3 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.

9.77.4.4 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.

9.77.4.5 QStringList QEPushButton::arguments [read, write]

Arguments for program specified in the 'program' property.

9.77.4.6 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'.

9.77.4.7 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.

9.77.4.8 QString QEPushButton::clickText [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.9 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

9.77.4.10 QString QEPushButton::confirmText [read, write]

Text used to confirm acion if confirmation dialog is presented

Reimplemented from QEGenericButton.

9.77.4.11 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.

9.77.4.12 QString QEPushButton::customisationName [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEGui can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI. Customisations are not applied if the GUI is opened as a dock.

Reimplemented from QEGenericButton.

9.77.4.13 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.

9.77.4.14 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).

9.77.4.15 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.

9.77.4.16 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.

9.77.4.17 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.

9.77.4.18 bool QEPushButton::leadingZero [read, write]

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

9.77.4.19 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"

9.77.4.20 Notations QEPushButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.77.4.21 QString QEPushButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.77.4.22 QPixmap QEPushButton::pixmap0 [read, write]

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

9.77.4.23 QPixmap QEPushButton::pixmap1 [read, write]

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

9.77.4.24 QPixmap QEPushButton::pixmap2 [read, write]

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

9.77.4.25 QPixmap QEPushButton::pixmap3 [read, write]

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

9.77.4.26 QPixmap QEPushButton::pixmap4 [read, write]

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

9.77.4.27 QPixmap QEPushButton::pixmap5 [read, write]

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

9.77.4.28 QPixmap QEPushButton::pixmap6 [read, write]

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

9.77.4.29 QPixmap QEPushButton::pixmap7 [read, write]

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

9.77.4.30 int QEPushButton::precision [read, write]

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

9.77.4.31 QString QEPushButton::pressText [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.32 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.

9.77.4.33 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

9.77.4.34 ProgramStartupOptionNames QEPushButton::programStartupOption [read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.77.4.35 QString QEPushButton::releaseText [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.36 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)

9.77.4.37 bool QEPushButton::trailingZeros [read, write]

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

9.77.4.38 UpdateOptions QEPushButton::updateOption [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.39 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.

9.77.4.40 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'.

9.77.4.41 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.

9.77.4.42 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.

9.77.4.43 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.

9.77.4.44 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'.

9.77.4.45 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.

9.77.4.46 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.

9.77.4.47 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.

9.77.4.48 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.

9.77.4.49 bool QEPushButton::writeOnClick [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.50 bool QEPushButton::writeOnPress [read, write]

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

Reimplemented from QEGenericButton.

9.77.4.51 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

9.78 QEPVNameLists Class Reference

Public Member Functions

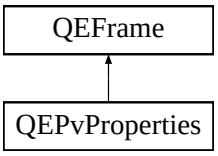
- void **prependOrMoveToFirst** (const QString &item)
- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

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

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

9.79 QEPvProperties Class Reference

Inheritance diagram for QEPvProperties:



Signals

- void **setCurrentBoxIndex** (int index)

Public Member Functions

- **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)
- qcaobject::QCaObject ∗ **createQcaItem** (unsigned int variableIndex)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **saveConfiguration** (PersistanceManager ∗pm)
- void **restoreConfiguration** (PersistanceManager ∗pm, restorePhases restorePhase)

- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

Properties

- QString variable
- QString variableSubstitutions

9.79.1 Property Documentation

9.79.1.1 QString QEPvProperties::variable [read, write]

EPICS variable name (CA PV)

9.79.1.2 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.

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

9.80 QEPvPropertiesManager Class Reference

Public Member Functions

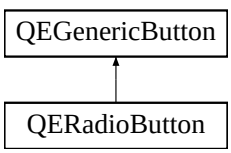
- **QEPvPropertiesManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget * **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

9.81 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, 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 ProgramStartupOptionNames { None = applicationLauncher::PSO_NONE, Terminal = applicationLauncher::PSO_TERMINAL, LogOutput = applicationLauncher::PSO_LOGOUTPUT, StdOutput = applicationLauncher::PSO_STDOUTPUT }
- enum CreationOptionNames {

  Open = QEActionRequests::OptionOpen, NewTab = QEActionRequests::OptionNewTab, NewWindow = QEActionRequests::OptionNewWindow, DockTop = QEActionRequests::OptionTopDockWindow,

  DockBottom = QEActionRequests::OptionBottomDockWindow, DockLeft = QEActionRequests::OptionLeftDockWindow, DockRight = QEActionRequests::OptionRightDockWindow, DockTopTabbed = QEActionRequests::OptionTopDockWindowTabbed,

  DockBottomTabbed = QEActionRequests::OptionBottomDockWindowTabbed, DockLeftTabbed = QEActionRequests::OptionLeftDockWindowTabbed, DockRightTabbed = QEActionRequests::OptionRightDockWindowTabbed, DockFloating = QEActionRequests::OptionFloatingDockWindow }

  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 requestAction (const QEActionRequests &request)

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 (const QEActionRequests &request)

  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)
- void programCompleted ()

  Program started by button has compelted.

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
- QString confirmText
- bool writeOnPress
- bool writeOnRelease
- bool writeOnClick
- QString pressText
- QString releaseText
- QString clickText
- QString clickCheckedText
- QString labelText
- QString program
- QStringList arguments

- ProgramStartupOptionNames programStartupOption
- QString guiFile
- CreationOptionNames creationOption
- QString prioritySubstitutions
- QString customisationName

9.81.1 Member Enumeration Documentation

9.81.1.1 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.

9.81.1.2 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.

DockTop Open new GUI in a top dock window.

DockBottom Open new GUI in a bottom dock window.

DockLeft Open new GUI in a left dock window.

DockRight Open new GUI in a right dock window.

DockTopTabbed Open new GUI in a top dock window (tabbed with any existing dock in that area)

DockBottomTabbed Open new GUI in a bottom dock window (tabbed with any existing dock in that area)

DockLeftTabbed Open new GUI in a left dock window (tabbed with any existing dock in that area)

DockRightTabbed Open new GUI in a right dock window (tabbed with any existing dock in that area)

DockFloating Open new GUI in a floating dock window.

9.81.1.3 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.

9.81.1.4 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.

9.81.1.5 enum QERadioButton::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a termainal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send doutput to standard output and standard error.

9.81.1.6 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)

9.81.1.7 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.

9.81.2 Constructor & Destructor Documentation

9.81.2.1 QERadioButton::QERadioButton (QWidget * *parent* = 0)

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

9.81.2.2 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.

9.81.3 Member Function Documentation

9.81.3.1 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)

9.81.3.2 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.

9.81.3.3 void QERadioButton::pressed (int *value*) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

9.81.3.4 void QERadioButton::released (int *value*) [signal]

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

9.81.3.5 void QERadioButton::requestAction (const QEActionRequests & *request*) [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.

9.81.4 Property Documentation

9.81.4.1 bool QERadioButton::addUnits [read, write]

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

9.81.4.2 Qt::Alignment QERadioButton::alignment [read, write]

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

9.81.4.3 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.

9.81.4.4 QStringList QERadioButton::arguments [read, write]

Arguments for program specified in the 'program' property.

9.81.4.5 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'.

9.81.4.6 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.

9.81.4.7 QString QERadioButton::clickText [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.8 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

9.81.4.9 QString QERadioButton::confirmText [read, write]

Text used to confirm acion if confirmation dialog is presented

Reimplemented from QEGenericButton.

9.81.4.10 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.

9.81.4.11 QString QERadioButton::customisationName [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEGui can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI.

Reimplemented from QEGenericButton.

9.81.4.12 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.

9.81.4.13 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).

9.81.4.14 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.

9.81.4.15 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.

9.81.4.16 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.

9.81.4.17 bool QERadioButton::leadingZero [read, write]

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

9.81.4.18 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"

9.81.4.19 Notations QERadioButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.81.4.20 QString QERadioButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.81.4.21 QPixmap QERadioButton::pixmap0 [read, write]

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

9.81.4.22 QPixmap QERadioButton::pixmap1 [read, write]

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

9.81.4.23 QPixmap QERadioButton::pixmap2 [read, write]

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

9.81.4.24 QPixmap QERadioButton::pixmap3 [read, write]

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

9.81.4.25 QPixmap QERadioButton::pixmap4 [read, write]

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

9.81.4.26 QPixmap QERadioButton::pixmap5 [read, write]

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

9.81.4.27 QPixmap QERadioButton::pixmap6 [read, write]

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

9.81.4.28 QPixmap QERadioButton::pixmap7 [read, write]

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

9.81.4.29 int QERadioButton::precision [read, write]

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

9.81.4.30 QString QERadioButton::pressText [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.31 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 substitttions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from QEGenericButton.

9.81.4.32 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

9.81.4.33 ProgramStartupOptionNames QERadioButton::programStartupOption [read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.81.4.34 QString QERadioButton::releaseText [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.35 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)

9.81.4.36 bool QERadioButton::trailingZeros [read, write]

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

9.81.4.37 UpdateOptions QERadioButton::updateOption [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.38 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.

9.81.4.39 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'.

9.81.4.40 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.

9.81.4.41 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.

9.81.4.42 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.

9.81.4.43 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'.

9.81.4.44 QString QERadioButton::variable [read, write]

EPICS variable name (CA PV)

9.81.4.45 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.

9.81.4.46 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.

9.81.4.47 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.

9.81.4.48 bool QERadioButton::writeOnClick [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.49 bool QERadioButton::writeOnPress [read, write]

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

Reimplemented from QEGenericButton.

9.81.4.50 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

9.82 QERecipe Class Reference

Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **optionsLayoutProperty** { **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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- bool **saveRecipeList** ()
- void **refreshRecipeList** ()
- void **refreshButton** ()
- void **userLevelChanged** (userLevelTypes::userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)

- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 **optionsLayout**
- 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**
- optionsLayoutProperty **optionsLayout**
- 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

9.83 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

9.84 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

9.85 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

9.86 QEScript Class Reference

`#include <QEScript.h>`

Public Types

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

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEScript** (QWidget ∗pParent=0)
- void **setShowScriptList** (bool pValue)
- bool **getShowScriptList** ()
- void **setShowNew** (bool pValue)
- bool **getShowNew** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setShowDelete** (bool pValue)
- bool **getShowDelete** ()
- void **setShowExecute** (bool pValue)
- bool **getShowExecute** ()
- void **setShowAbort** (bool pValue)
- bool **getShowAbort** ()
- void **setEditableTable** (bool pValue)
- bool **getEditableTable** ()
- void **setShowTable** (bool pValue)
- bool **getShowTable** ()
- void **setShowTableControl** (bool pValue)
- bool **getShowTableControl** ()
- void **setShowColumnNumber** (bool pValue)
- bool **getShowColumnNumber** ()
- void **setShowColumnEnable** (bool pValue)
- bool **getShowColumnEnable** ()
- void **setShowColumnProgram** (bool pValue)
- bool **getShowColumnProgram** ()
- void **setShowColumnParameters** (bool pValue)
- bool **getShowColumnParameters** ()
- void **setShowColumnWorkingDirectory** (bool pValue)
- bool **getShowColumnWorkingDirectory** ()
- void **setShowColumnTimeout** (bool pValue)
- bool **getShowColumnTimeout** ()
- void **setShowColumnStop** (bool pValue)
- bool **getShowColumnStop** ()
- void **setShowColumnLog** (bool pValue)
- bool **getShowColumnLog** ()
- void **setScriptType** (int pValue)
- int **getScriptType** ()
- void **setScriptFile** (QString pValue)
- QString **getScriptFile** ()
- void **setScriptText** (QString pValue)
- QString **getScriptText** ()
- void **setScriptDefault** (QString pValue)
- QString **getScriptDefault** ()
- void **setExecuteText** (QString pValue)
- QString **getExecuteText** ()

- void **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **insertRow** (bool pEnable, QString pProgram, QString pParameter, QString pWorkingDirectory, int pTimeOut, bool pStop, bool pLog)
- bool **saveScriptList** ()
- void **refreshScriptList** ()
- void **refreshWidgets** ()
- void **setScriptTypeProperty** (scriptTypesProperty pScriptType)
- scriptTypesProperty **getScriptTypeProperty** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 Attributes

- QComboBox * **qComboBoxScriptList**
- QPushButton * **qPushButtonNew**
- QPushButton * **qPushButtonSave**
- QPushButton * **qPushButtonDelete**
- QPushButton * **qPushButtonExecute**
- QPushButton * **qPushButtonAbort**
- QPushButton * **qPushButtonAdd**
- QPushButton * **qPushButtonRemove**
- QPushButton * **qPushButtonUp**
- QPushButton * **qPushButtonDown**
- QPushButton * **qPushButtonCopy**
- QPushButton * **qPushButtonPaste**
- _QTableWidgetScript * **qTableWidgetScript**
- QString scriptFile

  Define the file where to save the scripts (if not defined then the scripts will be saved in a file named "QEScript.xml")
- QString scriptText

  Define the XML text that contains the scripts.
- QString scriptDefault

Define the script (previously saved by the user) that will be load as the default script when the widget starts.

- int **scriptType**
- int **optionsLayout**
- QDomDocument **document**
- QString **filename**
- QList< _CopyPaste * > **copyPasteList**
- bool editableTable

  Enable/disable table edition.
- bool **isExecuting**

Properties

- bool showScriptList

  Show/hide combobox that contains the list of existing scripts created by the user.
- bool showNew

  Show/hide button to reset (initialize) the table that contains the sequence of programs to be executed.
- bool showSave

  Show/hide button to save/overwrite a new/existing script.
- bool showDelete

  Show/hide button to delete an existing script.
- bool showExecute

  Show/hide button to execute a sequence of programs.
- bool showAbort

  Show/hide button to abort the execution of a sequence of programs.
- bool showTable

  Show/hide table that contains a sequence of programs to be executed.
- bool showTableControl

  Show/hide the controls of the table that contains a sequence of programs to be executed.
- bool showColumnNumber

  Show/hide the column '#' that displays the sequential number of programs.
- bool showColumnEnable

  Show/hide the column 'Enable' that enables the execution of programs.
- bool showColumnProgram

  Show/hide the column 'Program' that contains the external programs to be executed.
- bool showColumnParameters

  Show/hide the column 'Parameters' that contains the parameters that are passed to external programs to be executed.
- bool showColumnWorkingDirectory

  Show/hide the column 'Directory' that defines the working directory to be used when external programs are executed.
- bool showColumnTimeout

Show/hide the column 'Timeout' that defines a time out period in seconds (if equal to 0 then the program runs until it finishes; otherwise if greater than 0 then the program will only run during this amount of seconds and will be aborted beyond this time)

- bool showColumnStop

  Show/hide the column 'Stop' that enables stopping the execution of subsequent programs when the current one exited with an error code different from 0.
- bool showColumnLog

  Show/hide the column 'Log' that enables the generation of log messages (these messages may be displayed using the QELog widget)
- scriptTypesProperty scriptType

  Select if the scripts are to be loaded/saved from an XML file or from an XML text.
- QString executeText

  Define the caption of the button responsible for starting the execution of external programs (if not defined then the caption will be "Execute")
- optionsLayoutProperty optionsLayout

  Change the order of the widgets. Valid orders are: TOP, BOTTOM, LEFT and RIG.
- bool variableAsToolTip
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool displayAlarmState

9.86.1 Detailed Description

This class is a EPICS aware widget. The QEScript widget allows the user to define a certain sequence of external programs to be executed. This sequence may be saved, modified or loaded for future usage.

9.86.2 Member Enumeration Documentation

9.86.2.1 enum QEScript::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.

9.86.3 Property Documentation

9.86.3.1 bool QEScript::allowDrop [read, write]

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

9.86.3.2 bool QEScript::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.

9.86.3.3 unsigned QEScript::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.

9.86.3.4 UserLevels QEScript::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'.

9.86.3.5 QString QEScript::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.

9.86.3.6 QString QEScript::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.

9.86.3.7 QString QEScript::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.

9.86.3.8 UserLevels QEScript::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 setUser-Level() 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'.

9.86.3.9 bool QEScript::variableAsToolTip [read, write]

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

9.86.3.10 bool QEScript::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/QEScript/QEScript.h
- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.87 QEShape Class Reference

`#include <QEShape.h>`

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

9.87.1 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.

9.87.2 Member Enumeration Documentation

9.87.2.1 enum QEShape::animationOptions

Options for how a variable will animate the shape.

9.87.2.2 enum QEShape::shapeOptions

Options for the type of shape.

9.87.2.3 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.

9.87.3 Constructor & Destructor Documentation

9.87.3.1 QEShape::QEShape (QWidget * *parent* = 0)

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

9.87.3.2 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.

9.87.4 Member Function Documentation

9.87.4.1 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.

9.87.4.2 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.

9.87.4.3 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.

9.87.4.4 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.

9.87.4.5 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.

9.87.4.6 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.

9.87.5 Property Documentation

9.87.5.1 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.

9.87.5.2 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.

9.87.5.3 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.

9.87.5.4 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.

9.87.5.5 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.

9.87.5.6 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.

9.87.5.7 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.

9.87.5.8 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'.

9.87.5.9 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'.

9.87.5.10 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'.

9.87.5.11 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'.

9.87.5.12 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'.

9.87.5.13 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'.

9.87.5.14 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'.

9.87.5.15 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'.

9.87.5.16 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'.

9.87.5.17 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'.

9.87.5.18 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.

9.87.5.19 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

9.87.5.20 double QEShape::offset1 [read, write]

Offset applied to data from the 1st variable before it is used to animate the shape

9.87.5.21 double QEShape::offset2 [read, write]

Offset applied to data from the 2nd variable before it is used to animate the shape

9.87.5.22 double QEShape::offset3 [read, write]

Offset applied to data from the 3rd variable before it is used to animate the shape

9.87.5.23 double QEShape::offset4 [read, write]

Offset applied to data from the 4th variable before it is used to animate the shape

9.87.5.24 double QEShape::offset5 [read, write]

Offset applied to data from the 5th variable before it is used to animate the shape

9.87.5.25 double QEShape::offset6 [read, write]

Offset applied to data from the 6th variable before it is used to animate the shape

9.87.5.26 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

9.87.5.27 QPoint QEShape::point10 [read, write]

10th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.28 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

9.87.5.29 QPoint QEShape::point3 [read, write]

3rd coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.30 QPoint QEShape::point4 [read, write]

4th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.31 QPoint QEShape::point5 [read, write]

5th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.32 QPoint QEShape::point6 [read, write]

6th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.33 QPoint QEShape::point7 [read, write]

7th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.34 QPoint QEShape::point8 [read, write]

8th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.35 QPoint QEShape::point9 [read, write]

9th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.87.5.36 double QEShape::scale2 [read, write]

Scale factor applied to data from the 2nd variable before it is used to animate the shape

9.87.5.37 double QEShape::scale3 [read, write]

Scale factor applied to data from the 3rd variable before it is used to animate the shape

9.87.5.38 double QEShape::scale4 [read, write]

Scale factor applied to data from the 4th variable before it is used to animate the shape

9.87.5.39 double QEShape::scale5 [read, write]

Scale factor applied to data from the 5th variable before it is used to animate the shape

9.87.5.40 double QEShape::scale6 [read, write]

Scale factor applied to data from the 6th variable before it is used to animate the shape

9.87.5.41 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'.

9.87.5.42 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.

9.87.5.43 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.

9.87.5.44 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.

9.87.5.45 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'.

9.87.5.46 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.

9.87.5.47 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.

9.87.5.48 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.

9.87.5.49 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.

9.87.5.50 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.

9.87.5.51 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.

9.87.5.52 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.

9.87.5.53 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.

9.87.5.54 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

9.88 QESlider Class Reference

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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

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

9.88.1 Member Enumeration Documentation

9.88.1.1 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.

9.88.2 Member Function Documentation

9.88.2.1 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.

9.88.3 Member Data Documentation

9.88.3.1 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).

9.88.4 Property Documentation

9.88.4.1 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.

9.88.4.2 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.

9.88.4.3 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.

9.88.4.4 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)

9.88.4.5 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'.

9.88.4.6 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.

9.88.4.7 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.

9.88.4.8 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.

9.88.4.9 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'.

9.88.4.10 QString QESlider::variable [read, write]

EPICS variable name (CA PV)

9.88.4.11 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.

9.88.4.12 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.

9.88.4.13 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

9.89 QESpinBox Class Reference

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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- QMenu ∗ **getDefaultContextMenu** ()

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**

9.89.1 Member Enumeration Documentation

9.89.1.1 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.

9.89.2 Member Function Documentation

9.89.2.1 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.

9.89.3 Property Documentation

9.89.3.1 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.

9.89.3.2 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.

9.89.3.3 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.

9.89.3.4 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)

9.89.3.5 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'.

9.89.3.6 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.

9.89.3.7 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.

9.89.3.8 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.

9.89.3.9 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'.

9.89.3.10 QString QESpinBox::variable [read, write]

EPICS variable name (CA PV)

9.89.3.11 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.

9.89.3.12 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.

9.89.3.13 bool QESpinBox::visible [read, write]

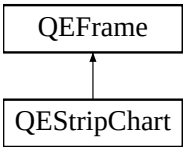
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

9.90 QEStripChart Class Reference

Inheritance diagram for QEStripChart:



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)

Protected Member Functions

- void **mousePressEvent** (QMouseEvent ∗event)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- void **setup** ()
- qcaobject::QCaObject ∗ **createQcaItem** (unsigned int variableIndex)
- void **establishConnection** (unsigned int variableIndex)
- void **saveConfiguration** (PersistanceManager ∗pm)
- void **restoreConfiguration** (PersistanceManager ∗pm, restorePhases restorePhase)
- void **addToPredefinedList** (const QString &pvName)
- QStringList **getPredefinedPVNameList** ()
- QString **getPredefinedItem** (int i)
- void **setRecalcIsRequired** ()
- void **setReplotIsRequired** ()
- void **evaluateAllowDrop** ()

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 QEStripChartItem

9.90.1 Property Documentation

9.90.1.1 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

9.91 QEStripChartAdjustPVDialog Class Reference

Public Member Functions

- **QEStripChartAdjustPVDialog** (QWidget ∗parent=0)
- void **setValueScaling** (const ValueScaling &valueScale)
- ValueScaling **getValueScaling** () const
- void **setSupport** (const double min, const double max, const QEDisplayRanges &loprHopr, const QEDisplayRanges &plotted, const QEDisplayRanges &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

9.92 QEStripChartContextMenu Class Reference

Signals

- void **contextMenuSelected** (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)

9.92.1 Constructor & Destructor Documentation

9.92.1.1 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

9.93 QEStripChartItem Class Reference

Signals

- void **itemContextMenuRequested** (const unsigned int, const QPoint &)
- void **requestAction** (const QEActionRequests &)

Public Member Functions

- **QEStripChartItem** (QEStripChart ∗chart, unsigned int slot, QWidget ∗parent)
- bool **isInUse** ()
- bool **isCalculation** ()
- void **setPvName** (QString pvName, QString substitutions)
- QString **getPvName** ()
- bool **isScaled** ()
- bool **getUseReceiveTime** ()
- QEArchiveInterface::How **getArchiveReadHow** ()
- QEStripChartNames::LineDrawModes **getLineDrawMode** ()
- void **setColour** (const QColor &colour)
- QColor **getColour** ()
- QEDisplayRanges **getLoprHopr** (bool doScale)
- QEDisplayRanges **getDisplayedMinMax** (bool doScale)
- QEDisplayRanges **getBufferedMinMax** (bool doScale)
- QCaDataPointList **determinePlotPoints** ()
- void **readArchive** ()
- void **normalise** ()
- void **plotData** ()
- 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

9.94 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_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
- static const int **NumberPrefefinedItems** = (SCCM_PREDEFINED_10 - SCCM_-PREDEFINED_01 + 1)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartNames.h

9.95 QEStripChartPushButtonSpecifications 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

9.96 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

9.97 QEStripChartState 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 files:

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

9.98 QEStripChartStateList Class Reference

Public Member Functions

- void **clear** ()
- void **push** (const QEStripChartState &state)
- bool **prev** (QEStripChartState &state)
- bool **next** (QEStripChartState &state)
- bool **prevAvailable** ()
- bool **nextAvailable** ()

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

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

9.99 QEStripChartStatistics Class Reference

Public Member Functions

- **QEStripChartStatistics** (const QString &pvName, const QString &egu, const QCaDataPointList &dataList, QEStripChartItem ∗owner, QWidget ∗parent=0)

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

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

9.100 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

9.101 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 **setYRangeStatus** (const QString &status)
- 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)

9.101.1 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

9.102 QESubstitutedLabel Class Reference

Public Member Functions

- **QESubstitutedLabel** (QWidget ∗parent=0)
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- void **setSubstitutionsProperty** (QString macroSubstitutionsIn)
- QString **getSubstitutionsProperty** ()
- QString **getLabelTextPropertyFormat** ()
- void **setLabelTextPropertyFormat** (QString labelTextIn)

Protected Attributes

- QString labelText

Properties

- QString textSubstitutions

9.102.1 Member Data Documentation

9.102.1.1 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

9.102.2 Property Documentation

9.102.2.1 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

9.103 recording Class Reference

Signals

- void **byteArrayChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **playingBack** (bool playing)

Public Member Functions

- **recording** (QWidget *parent=0)
- bool **isRecording** ()
- void **recordImage** (QByteArray image, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &time)
- void **nextFrameDue** ()

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

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

9.104 imageDisplayProperties::rgbPixel Struct Reference

Public Attributes

- unsigned char **p** [4]

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/brightnessContrast.h

9.105 screenSelectDialog Class Reference

Public Types

- enum **screens** { **PRIMARY_SCREEN** = -3, **THIS_SCREEN** = -2, **ALL_SCREENS** = -1 }

Public Member Functions

- **screenSelectDialog** (int numScreens, QWidget ∗parent=0)
- int **getScreenNum** ()

Static Public Member Functions

- static bool **getFullscreenGeometry** (QWidget ∗target, QRect &geom)

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

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

9.106 selectMenu Class Reference

Public Member Functions

- **selectMenu** (QWidget ∗parent=0)
- imageContextMenu::imageContextMenuOptions **getSelectOption** (const QPoint &pos)
- void **enable** (imageContextMenu::imageContextMenuOptions option, bool state)
- bool **isEnabled** (imageContextMenu::imageContextMenuOptions option)
- void **setChecked** (const int mode)
- void **setItemText** (imageContextMenu::imageContextMenuOptions option, QString title)

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

9.107 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

9.108 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

9.109 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

9.110 ValueScaling Class Reference

Public Member Functions

- void **reset** ()
- void **assign** (const ValueScaling &s)
- void **set** (const double dIn, const double mIn, const double cIn)

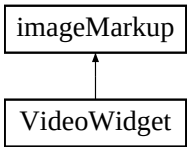
- void **get** (double &dOut, double &mOut, double &cOut) const
- void **map** (const double fromLower, const double fromUpper, const double toLower, const double toUpper)
- bool **isScaled** () const
- double **value** (const double x) const
- QEDisplayRanges **value** (const QEDisplayRanges &x) const
- void **saveConfiguration** (PMElement &parentElement) const
- 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

9.111 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)
- void **redraw** ()

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)
- QRect **scaleImageRectangle** (QRect r)
- int **scaleImageOrdinate** (int ord)

- QImage **getImage** ()
- QSize **getImageSize** ()
- bool **hasCurrentImage** ()
- void **markupChange** ()

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **mouseReleaseEvent** (QMouseEvent ∗event)
- void **mouseMoveEvent** (QMouseEvent ∗event)
- void **wheelEvent** (QWheelEvent ∗event)
- void **keyPressEvent** (QKeyEvent ∗event)
- void **markupChange** (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

9.112 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