logo

QE Framework –User Interface Design

Andrew Rhyder

28 November 2012

Contents

[Introduction 2](#_Toc341882786)

[Overview 2](#_Toc341882787)

[Qt Designer 2](#_Toc341882788)

[QEGui 2](#_Toc341882789)

[QE widgets 2](#_Toc341882790)

[GUI names 3](#_Toc341882791)

# Introduction

This document describes how to use the QE Framework to develop ‘code free’ Control GUI systems. It explains how features of the QE Framework widgets can be exploited, and how the QE Framework widgets interact with each other and with the QEGui application typically used to present the user interface.

While widget properties are referenced, a definitive list of the available properties is available in document QEReferenceManual.pdf.

This document is not intended to be a general style guide, or a guide on using Qt’s user interface development tool, Designer. Style issues should be resolved using facility based style guidelines, EPICS community standards, and general user interface style guides.

# Overview

In a typical configuration, Qt’s Designer is used to produce a set of Qt user interface files (.ui files) that implement an integrated GUI system. The QE Framework application QEGui is then used to present the set of .ui files to users. The set of.ui files may include custom and generic template forms, and forms can include nested sub forms. Other applications can also be integrated.

## Qt Designer

Designer is used to create Qt User Interfaces containing Qt Plugin widgets. The QE Framework contains a set of Qt Plugin widgets that enable the design of Control System GUIs. These are used, along with standard Qt widgets and other third party widgets.

## QEGui

QEGui is an application use to display Qt User Interface files (.ui files). Almost all of the functionality of a Control System GUI based on the QE Framework is implemented by the widgets in the user interface files. QEGui simply presents these user interface files in new windows, or new tabs, and provides support such as a window menu and application wide logging.

Simple but effective integration with Qt Designer is achieved with the option of launching Designer from the QEGui ‘Edit’ Menu. The user interface being viewed can then be modified, with the changes being automatically reloaded by QEGui.

# QE widgets

QE widgets are self contained. The application loading a user interface file – typically QEGui – does not have to be aware the user interface file even contains QE widgets. The Qt library locates the appropriate Plugin libraries that implement the widgets it finds in a user interface file.

While QE widgets need no support from the application loading the user interface containing them, some QE widgets are capable of interacting with the application, and other widgets. For example, a QEPushButton widget can request that whatever application has loaded it open another user interface in a new window.

QE widgets fall into two categories:

* EPICS aware Qt Widgets. These widgets are based on a standard Qt widget and generally allow the widget to write and read data to a control system. For example, QELabel is based on QLabel and displays data updates as text.
* Control System Widgets. These widgets are not readily identifiable as a single standard Qt widget and implement functionality specific to Control systems. For example, QEPlot displays waveforms.

# GUI names

The QEGui application reads the windowTitle property of the top level widget in a user interface file. It then applies any macro substitutions to the name and uses it as the GUI title. Figure 1 shows a windowTitle property that includes macros being edited in Designer, with the same user interface being displayed by QEGui with the appropriate macro substitution.



Figure windowTitle Property