QE Framework -Getting Started

Andrew Rhyder

21 March 2014

Contents

Introduction

This document contains details on configuring a run time environment or an application development environment where the QE Framework will be used.

The QE Framework can be used in three ways:

- **Code Free GUI systems** using Qt's Designer application with the QE Framework plugins to design GUIs, and the QEGui application to present GUIs to users.
- **Code Rich GUI development** using Qt's Integrated Development Environment with the QE Framework widgets and data objects to design GUI applications.
- **Console application development** using Qt's Integrated Development Environment with the QE Framework data objects to design console applications that can access EPICS data.

Note, there are many variations to the above, such as using another Integrated Development Environment like Eclipse, or developing new plugin widgets to implement desired functionality, then using those widgets within a code free GUI development.

Other documents you may be interested in are:

• QEReferenceManual.pdf	QE framework class references. Includes Plugin Property descriptions
• QEFrameworkOverview.docx	Overview of the Framework components and typical usage paradigms
• QEGuiGuide.doc	QEGui is an application that presents a set of Qt User Interface files as an integrated Control System GUI. It is the core component in a 'code free' control GUI solution, along with QT's Designer which is used to lay out the user interface forms.

How to use this document

You may first like to read 'Components and prerequisites' (page 2) to familiarise yourself with the QE framework components and what may be required to use the QE framework.

Then refer to 'Setting up your environment' (page 3) to determine what is needed to support your specific tasks.

Once clear about what you require, follow the specific instructions referenced.

Note, this document is Linux centric. The QE Framework is, however, platform independent and has been build and run under windows. This document is also applicable for windows, although the specific commands will require interpretation.

Components and prerequisites

The QE framework includes the following components:

- QE library (libQEPlugin.so or QEPlugin.dll)
  This library contains all classes that implement the QE framework including data objects, widgets (Qt plugins) and supporting classes.
- QEGui
  A stand alone application that which is used to present a collection of Qt User Interface files as an integrated control centric GUI system.
- Documentation
  Includes:
  - Getting started guide (this document)
  - User manual
  - QE framework reference

The following components are required to use the QE framework. Some are optional, depending on what you are doing:

- Qt libraries.
  The C++ based application framework underpinning QE.
  At the time of writing, the QE Framework is being used with Qt 4.6 to Qt 4.8. Pre Qt 4.6 versions have been used and are likely to still be OK.
- Qwt library
  'Qt Widgets for Technical Applications' provides the base widgets used for the plotting and tracing widgets in the QE framework
- FFmpeg library
  'FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video.' FFmpeg can be used by the QEImage widget to provide an MPEG stream image source. The FFmpeg library is optional and is not used by the QEImage widget by default.
- CA libraries
  EPICS Channel Access provides access to EPICS
- Gnu tools (Optional)
  Qt uses gcc and gmake to build the QE framework.
- Qt designer (Optional)
  If you are creating Qt User Interface files for a 'code free' GUI system, you will also need Qt Designer for laying out the user interface files.
- Qt Creator (Optional)
  If you are developing applications that use the QE framework, you will also need the Qt Creator IDE.

Setting up your environment

Depending on what you are going to do with the QE framework, your set up tasks may vary.

No matter what you are doing…

No matter what you are doing, you will require:

- The Qt 4 libraries. You may well already have Qt 4 as it is used by many other common applications. Refer to 'Qt libraries' (page 4) for details.
- The QE framework. Refer to 'QE framework' (page 5) for details.

Code free GUI implementation.

Nothing extra required. Use the QEGui application (part of the QE framework package) to present a control system GUI fully defined by a suite of Qt User Interface files.

Refer to the QEGui documentation for details.

Code free GUI development

If you want to create a control system GUI defined by a suite of Qt User Interface files, you require:

- Qt Designer. This is Qt's drag and drop form design tool. It is included with the Qt 4 libraries and with the Qt SDK

Code rich development

If you want to write graphical or non-graphical application code using the QE framework, you will require:

- A development environment such as Qt's Integrated Development Environment 'Creator'. Refer to http://qt-project.org/ and follow the links to Qt Creator for details.

Note, there are alternatives to Creator, such as the Eclipse IDE or your favourite text editor plus various Qt and Gnu command line tools. For details, see elsewhere.

Installation tasks

Qt libraries

Firstly, you may already have the Qt libraries as many common applications depend on them. To check on Linux run the command:

```
yum info qt-x11
```

You will also end up with the Qt libraries if you install the Qt SDK.

If you don't already have the libraries, and you don't intend on installing the Qt SDK, you can install the Qt libraries with the following command:

```
yum install qt-x11
```

Alternatively, the Qt libraries (and instructions to build them) are available for download from http://qt-project.org/downloads.

Note, at the time of writing, the QE Framework is being developed in Qt environments from Qt 4.6 to Qt 4.8. Pre Qt 4.6 versions have been used and are likely to still be OK.

Note, the Qt library dependencies should be managed automatically when installing from a Linux distribution repository. The dependencies may not be accommodated when downloading and building Qt manually. One common dependency problem has been older versions of freetype and fontconfig on RedHat EL 5. Installing freetype-2.4.3 and fontconfig-2.8.0 on RedHat EL 5 resolves this dependency issue.

QE framework

The QE Framework is available at http://sourceforge.net/projects/epicsqt

To use the framework, you must download the epicsqt project, build it, and deploy the appropriate QE Framework components.

1. **Download the epicsqt project**

   There are three ways described below to get the framework:

   - Download the framework source in a single gzip file
   - Use an svn client to extract the source from its svn repository
   - Use an svn client to download a single framework makefile which can then be used to download the framework.

2. **Build the epicsqt project**

   There are three ways described below to build the QE Framework:

   - Qt Creator
   - qmake and make commands
   - Framework makefile

   Note, no matter how you build the framework QWT will need to be accessible. If installed correctly - QMAKEFEATURES includes the location of the file qwt.prf - the 'CONFIG += qwt' statement in the project file will be all you need. As an alternative, you will need to ensure the QWT libraries can be located for linking, and you will need to define an environment variable to point to the QWT include files. The following is a typical example of the command required:

```
export QWT_INCLUDE_PATH=/usr/include/qwt
```

   Note, no matter how you build the framework FFmpeg will need to be available if you want the QEImage widget to be able to stream MPEG into the widget. If you want to be able to use an MPEG stream in the QEImage widget, install FFmpeg and define the environment variable QE_USE_MPEG:

```
export QE_USE_MPEG=YES
```

If not using FFmpeg ensure the environment variable QE_USE_MPEG is not defined at all. When expecting the FFmpeg library the project file framework.pro will look for the FFmpeg libraries and include files in their default locations.

3. **Deploy the appropriate QE Framework components**

There are hundreds of ways to deploy the QT based libraries and applications. Two simple alternatives are described below. Both enable the QEGui application and allow the QE Framework to be used as a library loaded by applications and as a Qt Plugin:

- Manual deployment
- Framework makefile

Note, no matter how you deploy the framework, it may need to locate a Channel Access archiver and will attempt to do so using the environment variable QE_ARCHIVE_LIST. The following is a typical example of the command required to define this variable:

```
export QE_ARCHIVE_LIST=archiver.synchrotron.org.au:80/cgi-
bin/ArchiveDataServer1.cgi archiver.synchrotron.org.au:80/cgi-
bin/ArchiveDataServer2.cgi
```

Note, all the examples for getting and building the QE Framework assume the epicsqt project has been placed in the directory ~/epicsqt.

Getting the epicsqt project as a gzip file

The latest gzip file is available for download from http://sourceforge.net/projects/epicsqt.

The same gzip file, along with earlier versions and links to documentation is available at http://sourceforge.net/projects/epicsqt/files/

The contents of the gzip can be extracted using the following command:

```
tar -zxvf epicsqt-1.1.8-src.tar.gz
```

Getting the epicsqt project using svn

The latest source code for the QE Framework can be downloaded using the following svn command:

```
svn co http://svn.code.sf.net/p/epicsqt/code/trunk epicsqt/trunk
```

This command will create a directory 'epicsqt/trunk' containing the QE Framework project.

An svn client can also be used to get earlier version of the source code.

SourceForge also provide a browser based interface to the svn repository at http://epicsqt.svn.sourceforge.net/viewvc/epicsqt/

Getting the build script using svn

A single build script that will download the entire project source from the svn repository can be obtained using the following svn command:

```
svn export http://svn.code.sf.net/p/epicsqt/code/trunk/resources/makefile
```

The following command will download (checkout) the QE Framework:

```
make checkout
```

Note, the makefile can also be used to build, clean, and package the QE Framework, as well as create a RPM package and upload it into SourceForge download area. Refer to 'Building the framework using the framework makefile' (page 7) and 'Deploying the QE Framework – using framework makefile' (page 8) for details.

Note, the makefile is itself part of the epicsqt project files and can be found in the 'resources' directory.

Building the framework using Qt Creator

- Open the epicsqt.pro file in Qt Creator. The epicsqt.pro file is in the top level directory of the epicsqt project.
- Ensure shadow building is unchecked in the 'Projects' options.
- Build the project.

If you are using a multi core processor, adding a compile option –j*n* (where *n* is the number or concurrent compilations) will speed up compilation considerably. For example, -j4 uses most of the CPU time available on a 4 core CPU.

Building the framework using qmake

To build the QE Framework using Qt's qmake, enter the following commands in the epicsqt project directory:

- qmake
- make

Building the framework using the framework makefile

A single project makefile is available to download the entire project source from the svn repository, build the framework, package the framework into a self-contained directory and, eventually, create a RPM file containing the framework. The makefile can be downloaded from the svn repository - refer to 'Getting the build script using svn' (page 6) for details - and is also itself part of the epicsqt project source and can be found in the 'resources' directory.

The following targets can be specified for the project makefile:

```
make                # (default) download and build the framework
make checkout       # download the latest version of the framework (i.e. from the "trunk")
make tag=XXX        # download tag XXX of the framework (i.e. from the "tags")
make framework      # build the framework
make clean          # clean the framework
make package        # copy all deployable components into a package directory
make rpm            # create a RPM file containing the framework
make upload_rpm     # upload RPM file into SourceForge download area
```

Deploying the QE Framework – Manual deployment

There are many options for deploying the QE Framework, but the following describes a simple deployment that allows the QE Framework applications to be located and the QE Framework library to be used as both a standard library and a Qt Designer Plugin.

Extend the path to include the location of the QEGui application:

```
export PATH=$PATH:/home/<user>/epicsqt/trunk/applications/QEGuiApp
```

Ensure applications can find the QE Framework library as a standard library:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/<user>/epicsqt/trunk/framework/designer
```

Ensure Qt can find the QE Framework library as a Qt Designer Plugin:

```
export QT_PLUGIN_PATH=:/home/<user>/epicsqt/trunk/framework
```

Note, the above command allows Qt to find a 'designer' directory containing the QE Framework library libQEPlugin.so. Refer to Qt documentation on the many ways Qt can locate and load Plugins.

Deploying the QE Framework – using framework makefile

A single project makefile is available to download the entire project source from the svn repository, build the framework, and package the framework. The makefile can be downloaded from the svn repository - refer to 'Getting the build script using svn' (page 6) for details - and is also itself part of the epicsqt project source and can be found in the 'resources' directory.

The following command uses the framework makefile to place all required applications and libraries (including CA libraries) into a single 'package' directory, and to set up environment variables to allow the QE Framework library to be located by applications and by Qt's Plugin system:

```
make package
```

Writing applications that use QE framework classes

When writing applications that link to QE classes, including QE widgets and data objects, the following line should be included in the Qt project file:

LIBS += -L/home/<*user*>/epicsqt/trunk/framework/designer –lQEPlugin

Using the QE Framework as a Qt Plugin library

The QE framework library can be used as a Qt Plugin library in the following scenarios:

- Laying out forms within Qt creator (Designer embedded within Qt Creator)
- Laying out forms within Designer
- Loading Qt .ui files by any application, including the QEGui application which is part of the QE Framework package. Any application that dynamically loads .ui files (using QUiLoader) is

relying on the Qt libraries to load the appropriate plugin libraries to support the .ui file. This will include the QE Framework library when QE widgets are included in the .ui file.

The deployment instructions in this document will ensure the QE Framework can be located as a Qt Plugin by any application including Qt Creator and Designer. There are other ways of setting up a plugin to be located by the appropriate applications. Refer to Qt's Plugin documentation for details.