KSTAR WIDGET TOOLKIT USING QT LIBRARY FOR THE EPICS-BASED CONTROL SYSTEM

S. Baek, S. Lee, M.K. Park, H.K. Na, and M. Kwon,
NFRI, 113 Kwahangno, Yuseong-gu, Daejeon, 305-333, KOREA

Abstract

The KSTAR Widget Toolkit (KWT) was developed as a development toolkit of channel access (CA) client application for the KSTAR commissioning. The KWT is based on Qt library and includes channel access interface to communicate with EPICS. In order to enhance development speed and increase aesthetic quality of application, 18 plug-in widgets were developed to enable for developers to create new panel using drag and drop method. Some of them use QWT as a plotting library and some widgets display alarm status with a specified color according to the EPICS alarm convention. The KWT has cross-platform development environment and feasibility of extending new widgets using Qt plug-in API with plenty of documents and tutorials. Around 120 panels and several applications such as multi-channel plotting tool, process variable searching tool, and logbook application were developed through the KWT and they proved functionality of the KWT being used for the integrated control and machine control during the KSTAR commissioning. The KWT is applicable to fast and easy development of operator interfaces and applications for the EPICS-based control system.

INTRODUCTION

The KWT was developed as a development toolkit of Experimental Physics and Industrial Control System (EPICS [1]) CA client application for the Korea Superconducting Tokamak Advanced Research (KSTAR [2]) commissioning. It's Qt widget library which includes Qt-CA (Channel Access) interface and 18 plug-in widgets. CA is the software component that allows a Channel Access client application to access control-system data which may be located on different hosts throughout a network [3]. This paper explains the principal of KWT and application of it. We assume that readers are familiar with EPICS.

DEVELOPMENT STATUS

Items listed below show the requirements for the KSTAR Operator Interfaces (OPIs) or the development toolkit of them.

- Performance
- Stable EPICS CA communication
- Easy & fast development
- Maintenance
- Usability
- Consistency of appearance

Qt was selected as a graphic library to develop KSTAR OPI development toolkit because of its plug-in property, cross-platform development environment, good appearance, abundant documents, and flexible license policy.

Features of KWT Library

KWT inherits the features listed below from Qt library. Regarding to the portability, it uses Qt cross-platform development environment but Qt-CA interface was not fully tested on platforms except linux. For the integrated development tools, it's possible to use Qt designer, which is a Qt cross-platform integrated development environment (IDE), to make OPI panels because KWT widgets were designed as custom plug-in widgets for it.

- Intuitive C++ class library
- Portability across desktop and embedded operating systems
- Integrated development tools with cross-platform IDE
- High runtime performance and small footprint on embedded [4]

Inheritance Hierarchy of Library Properties

KWT libraries depend on Qt-4.3.2 and some plotting widgets among them inherit properties from QWT libraries. In addition, to communicate with EPICS, Qt-CA library in KWT interfaces with EPICS base-3.4.18.2. Some classes use boost library as a Standard Template Library support package. The libraries listed below are prerequisite libraries for the KWT installation and inheritance hierarchy of the KWT is shown in Fig. 1 in detail.

- Qt -4.3.2
- QWT-5.0.0rc1
- EPICS base-3.14.8.2
- Boost library

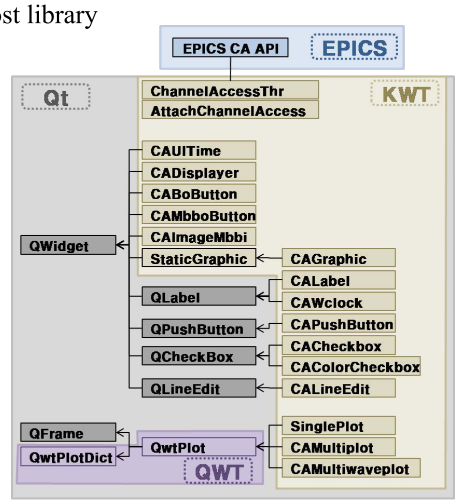

Figure 1: Inheritance hierarchy of the KWT.

Principal of the Qt-CA Interface

The core libraries of KWT Qt-CA interface are AttachChannelAccess library and ChannelAccessThr library. Working procedure of AttachChannelAccess is drawn in Fig. 2 briefly. Usually this library is used for a widget containing child widgets of a UI file created by Qt designer. After initialization, this instance acquires list of CAobjects from the QWidget or UI file. For every CAobjects it links event filter and work thread. If the CAobject has control property, it is registered as control object. The work thread created using ChannelAccessThr class and it updates all hash tables for all CAobjects. Hash table structure named by ChAccess which is updated by work thread is shown in Fig. 2. It is updated at every pre-defined period or CA monitor event.

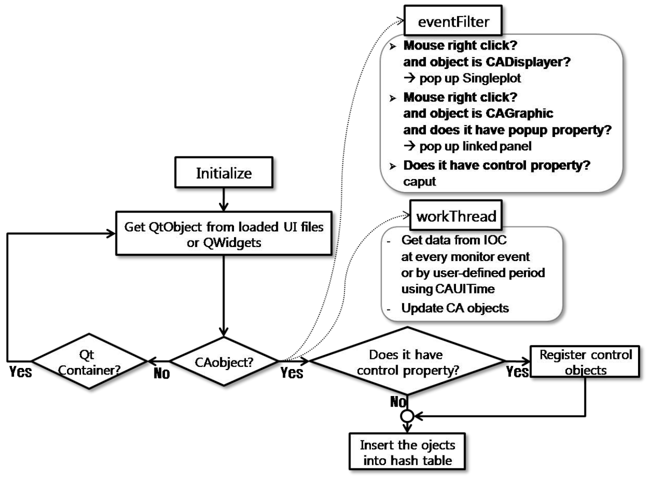
eventFilter
Mouse right click? and object is CADisplayer? → pop up Singleplot
Mouse right click? and object is CAGraphic and does it have popup property? → pop up linked panel
Does it have control property? caput
Initialize
Get QtObject from loaded UI files or QWidgets
workThread
- Get data from IOC at every monitor event or by user-defined period using CAUITime
- Update CA objects
Qt Container?
Yes
No
CAobject?
Yes
Does it have control property?
Yes
Register control objects
No
Insert the ojects into hash table

Figure 2: Working procedure of AttachChannelAccess library.

KWT Widgets

The KWT includes 18 widget classes and brief explanations about the widgets are as follows.

AttachChannelAccess class attaches a widget or an UI file, which is created using Qt designer, to a structured hash table vector. The widget is usually a container having lots of KWT widgets. The hash tables are updated by the ChannelAccessThr by the pre-defined refresh policy.

ChannelAccessThr class updates the hash tables, which are created by AttachChannelAccess class, with the latest CA results.

CAUITime class defines refresh policy of a widget to be attached to a structured hash table vector by the AttachChannelAccess. Refresh policy may be periodic or event-driven. If the refresh policy is periodic, a period by second should be defined additionally. If it is event-driven, the parent widget updates its children at every CA monitor event. CAUITime has another unique property, control master. If a developer defines control master process variable (PV) in the CAUITime instance, all control widgets' control property may be locked by the control master PV.

CADisplayer displays numeric data with the corresponding alarm information with a specific color. MAJOR, MINOR, NO_ALARM, or INVALID alarms from EPICS Input Output Controller (IOC) changes CADisplayer fill-color with red, yellow, dark green, or grey each other. In a case with CA disconnection its fill-color is changed to white showing disconnection message. If an operator clicks the instance with right mouse button, a SinglePlot instance with the corresponding PV is popped up.

CABoButton is a pair of QPushButtons with a false button and a true button. If an operator pushes one of them, it sends operator's choice to the PV. A developer can make it be committed by the matched password or confirmation.

CAMbboButton is a collection of multiple CAPushButtons for EPICS mbbo record. If an operator pushes one of them, it sends the corresponding numeric value to the PV..

CAImageMbbi is a collection of multiple QLabels for EPICS mbbi record. Particularly, QLabels in it are displayed with pixmap images. In a case with CA disconnection, a white pixmap image is displayed.

StaticGraphic is a symbol library. Even if it is included in the KWT library, it doesn't communicate with EPICS. A developer can choose one of the pre-defined geometries. Various geometries including vacuum devices, arrows, ellipse, rectangle, etc are already pre-defined. It is useful to make a simple diagram or legend. **CAGrahpic** inherits properties from **StaticGraphic** class and changes its color by the PV's alarm status or value. If a developer adds a new geometry to StaticGraphic class, CAGraphic instance gets additional geometry automatically.

CALabel was designed to display text label corresponding to the value. But it became diversified to display integer value, double value or text label by the property.

CAWclock displays time information as text label. To display the right time information you should install timestamp library with it.

CAPushButton is a QPushButton which sends operator's command to the PV. It has 4 button types consists of toggle-button, true-button, false-button, and pulse-button.

CAColorCheckbox is used for CAMultiplot or CAMultiwaveplot widget. It displays the color information which is sent by CAMultiplot or CAMultiwaveplot widget on its text label.

CALineEdit is a QLineEdit which sends entered numeric data to the PV. It is usually used for EPICS analogue output record.

SinglePlot is a single channel plotting class using QWTPlot library. It is linked to the CADisplayer instance.

CAMultiplot is a multi-channel plotting class using QWTPlot library. It can plot 10 analogue data at once to the limit. CAColorCheckbox is used to show PV names for each colored lines. **CAMultiwaveplot** is a multi-channel plotting class using QWTPlot library for the

EPICS waveform record and its usage is similar to CAMultiplot widget.

OPI Development Example with the KWT

Widgets in the KWT were designed as plug-in widgets to be inserted using Qt designer. Below procedures show a simple example using the KWT.

- Create a Widget using Qt designer.
- Drag a CADisplayer instance from KSTAR widget box and drop it to the proper position.
- Configure it with a valid PV name at the pvname property.
- Save the widget as uifilename.ui.
- Develop a simple main application (main.cpp) referring to Table 1.
- Compile the main application.
- Execute it.

Table 1: A simple Example of Main Application

```
#include <QtUiTools>
#include <QtGui>
#include "qtchaccesslib.h"

int main (int argc, char *argv[])
{
   QApplication app(argc, argv);
    AttachChannelAccess attach("uifilename.ui, 1);
   QWidget *pwidget = attach.GetWidget();
   Pwidget->show();
   Return app.exec();
}
```

APPLICATIONS OF KWT TO KSTAR

KSTAR OPI Panels

Over than 120 OPI panels were developed using KWT and about 50 panels were developed using other EPICS extensions such as QtCAtool, MEDM, and EDM. Some screenshots of the KSTAR OPIs are shown in Fig. 3. During the KSTAR commissioning, they were used for remote operation without any serious problems. The OPI panels with KWT were well accepted by the operators because of the simple panel switching and consistent appearance.

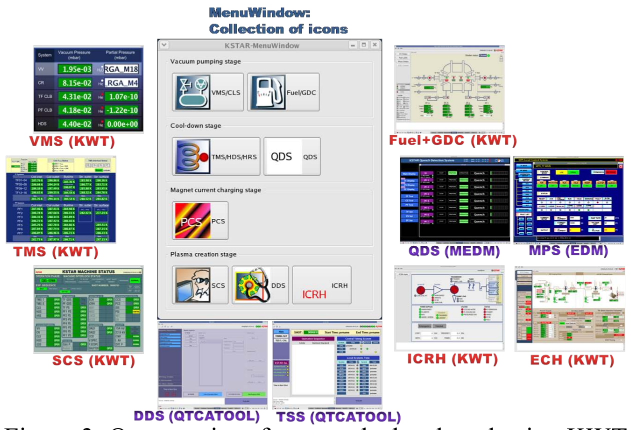

Figure 3: Operator interface panels developed using KWT library and EPICS extensions.

Besides OPI panels, some applications such as multi-channel plotting tool, process variable searching tool, and logbook application were developed using KWT library.

Known Problems

The most serious problem of the KWT OPIs discovered during the KSTAR commissioning was abnormal stop at some CA server down. Even if the exact reason of it was not clarified yet, it happened with the CA server down having multiple IOCs in it. Total number of abnormal stop of KWT OPIs was less than 5 times during the KSTAR commissioning.

LICENCE POLICY AND S/W RELEASE

KWT is available as free software under the GNU General Public License (GPL). The source code for the KWT will be released on SourceForge in the CVS repository of the KWT project [5].

CONCLUSION

Even if it took more time and effort to develop not only OPI panels but also development toolkit, KSTAR OPIs developed using KWT library satisfied almost requirements concerning performance, easy & fast development, nice maintenance, usability, and consistency of appearance. However, abnormal stop occurred intermittently at some CA server down should be fixed as soon as possible to enhance stability of CA communication.

REFERENCES

[1] http://www.aps.anl.gov/epics .
[2] M. Kwon et al., "The control system of KSTAR", Fusion Engineering and Design, June 2004, Volume 71, Issues 1-4, Pages 17-21.
[3] http://lansce.lanl.gov/EPICSdata/ca/client/caX5Ftutor-4.html#HEADING4-0.
[4] http://qt.nokia.com/.
[5] http://kwt.sourceforge.net.