DEVELOPMENT OF THE PYTHON/TK WIDGETS FOR THE CONTROL SYSTEM BASED ON EPICS

T. T. Nakamura, T. Katoh, N. Yamamoto, KEK, Tsukuba, Japan

Abstract

During the commissioning of accelerators, many application programs are built and some of them are scrapped every day. Quick development of a large amount of application software and keeping their quality high are essential for the efficient operation and machine studies of the accelerators. Especially, to construct excellent GUI (graphical user interface) is key technology. EPICS (Experimental Physics and Industrial Control System)[1] has several graphical editors to develop GUI. Although they are easy to use, they are not so flexible to use customized widgets. On the other hand, development of GUI by programming languages provides full flexibility. Python is a simple but powerful programming language suitable for the quick development of the application programs. Combining Python with Tcl/Tk widget set is one of the most powerful tools. In KEKB accelerators control system, several widgets for controls have been developed. They are not only for EPICS, but also general purpose. Using object-oriented feature of Python, these widgets have been easily customized for EPICS. Not only these widgets are presented, but the framework of the development of them is also discussed.

1 INTRODUCTION

In KEKB accelerators control system[1], EPICS has been used as the core software component. The basic components of EPICS are OPI (Operator Interface, which is a UNIX based workstation), IOC (Input/Output Controller, which is VME/VXI based chassis containing a processor) and network. In the IOC there is a memory resident database. A PV (Process Variable) is an atomic component of the database. EPICS provides CA (Channel Access) protocol, which provides network transparent access to IOC database based on client/server model. The PV accessed through CA is sometimes called "channel".

In EPICS standard distribution, there are several kinds of graphical editors to build application programs with GUI. Although they are easy to use and powerful for simple operations, they have neither enough flexibility to use user-customized widgets nor programming capability to implement complex control logics.

In KEKB accelerators control system, we have been extensively using Python as a programming language to build application program on OPI. Python is an easy to learn, interpretive programming language. It is not only simple but also has powerful features: efficient high-level data structures, object-oriented programming and rich libraries, which cover wide range of areas. They make it powerful tool for rapid application development. Python has some GUI extension modules. Among them Tkinter is most popular. "Tkinter" is an interface module to Tcl/Tk, with which Python gains ability to build GUI using Tk widgets. We have also developed "ca" module, which is an interface module to EPICS CA.

Although the combination of Tkinter and ca module has powerful capability to build any kind of control application programs, most of the non-expert programmers still feel complexity about GUI programming. Thus we decided to develop a ready-made widget library convenient to build control application programs.

The development policies we adopted are as follows.

(1) Effort of application programmer should be minimized.

(2) Development of the widgets should be done not only for EPICS system but also adaptable to other purposes. In other words, general-purpose widget could be reusable also for EPICS system.

(3) The mechanism to introduce CA capability into the general-purpose widget should be simple, systematic and flexible.

Following sections show how we have designed the framework of the widget library development.

2 DESIGN OF CAVAR CLASS

2.1 Basic design of caVar class

Some Tk widgets accept a variable to configure itself dynamically. The variable is implemented in Tcl and accessible through several classes that are derived from "Variable" class in Tkinter. We introduced special class named "caVar", which is derived from the Variable class in Tkinter. The caVar class provides special functions to communicate with EPICS PV through CA. From caVar class we define four derived classes: caStringVar, caIntVar, caDoubleVar and caBooleanVar. We call the instance of these four classes "CA-variable" for simplicity. caVar class has "assign" method, which associates the CA-variable with a channel specified by the channel name. The assignment is done in one of the

[1] http://epics.aps.anl.gov/asd/controls/epics/EpicsDocumentation/WWWPages/

three modes: “monitor mode”, “put mode” and “both mode”.

In the “monitor mode”, the assigned channel is monitored and the value of the CA-variable is automatically updated whenever the value of the channel is changed. In the “put mode”, whenever the value of the CA-variable is modified, the new value is automatically put to the associated channel. In the “both mode” the CA-variable has same capability as in the monitor mode. In addition to the monitoring, its “set” method puts a value to the associated channel.

Programming using caVar class takes following two steps. First, create CA-variable and assign a channel to it. Then, pass it to the general-purpose widget that can accept a variable. Since all of the EPICS specific features are confined in caVar class, the development of a new widget has little constraint. The only requirement is that the widget must be designed to accept a variable as usual Tk widgets do.

2.2 Additional features of caVar class

The constructor of the caVar class accepts some optional arguments.

Using the “conv” option, a conversion function can be specified, which introduces data conversion between the CA-variable and the channel. In the “monitor mode” or the “both mode”, the value from the channel is converted by this function, and then set to the CA-variable. In the “put mode”, the value of the CA-variable is converted by this function, and then put to the channel.

Using “format” option, a format string with C sprintf()-style can be specified, which is used for the conversion from any type to string type. This special conversion is performed after the conversion by the “conv” option if specified.

A conversion function for the set method in the “both mode” can be also specified by “rconv” option.

3 DESIGN OF CAWIDGET CLASS

3.1 Basic design of caWiget class

Although using the general-purpose widget together with CA-variable is simple and flexible way, there is still some room to decrease amount of coding for non-expert programmers. They prefer to create a widget instance by single function call. Thus we also decided to develop “CA-widgets”, which are the customized version of the widgets for EPICS CA. To develop the CA-widgets, we introduced “caWidget” class. The caWidget is the abstracted class that manages CA related functions. Its constructor creates a CA-variable and assigns the channel automatically.

The customizing procedure is very simple by using multiple-inheritance feature of Python. (1) Define the customized class as derived class from both caWidget class and the original widget class. (2) Define several (typically four) class constants. (3) Define some methods to override original ones if necessary. Method redefinition is usually not necessary even for the constructor method.

3.2 Additional features of caWidget class

caWidget class provides some additional features to the original widget: Channel Information Display (CID) and Dynamic Configuration Rule (DCR).

While middle mouse button is pressed on the widget, CID appears. The CID is a small window that contains some information about the assigned channel. This mechanism is automatically introduced when the widget is defined as the derived class of caWidget. Figure 1 shows an example of a CID.

Each PV has severity, which indicates that the PV is in some abnormal states or not. To warn operators about such an abnormal state, it is desirable to change colour or some appearance of the widget according to the severity. DCR provides such capability. The widget may change its configuration dynamically by the severity of the assigned channel. The CA-widget can define default rule that maps each severity into the configuration. Also user can define his/her own rule with “rule” option of the widget constructor. Figure 1 shows an example of the widget turned red by the MAJOR severity. In addition to the severity, the configuration rule when CA is disconnected can be defined.

4 EXAMPLES

We have developed several widgets for the control system. We have also developed customized version of them for EPICS using caWidget. Following subsections show some of them as examples. Some widgets in Tkinter such as Label, Checkbutton and Radiobutton have been also customized using caWidget.

4.1 caBitPattern widget

Figure 1 shows four examples of caBitPattern widget. Each bit of the integer value of the channel is shown by a small indicator with circular or rectangular shape.

4.2 caWritableLabel widget

Figure 2 shows an example of caWritableLabel widget. It shows the value of the assigned channel as a character string. When the left mouse button is double-clicked on the widget, a dialog box appears to let user input a new value.

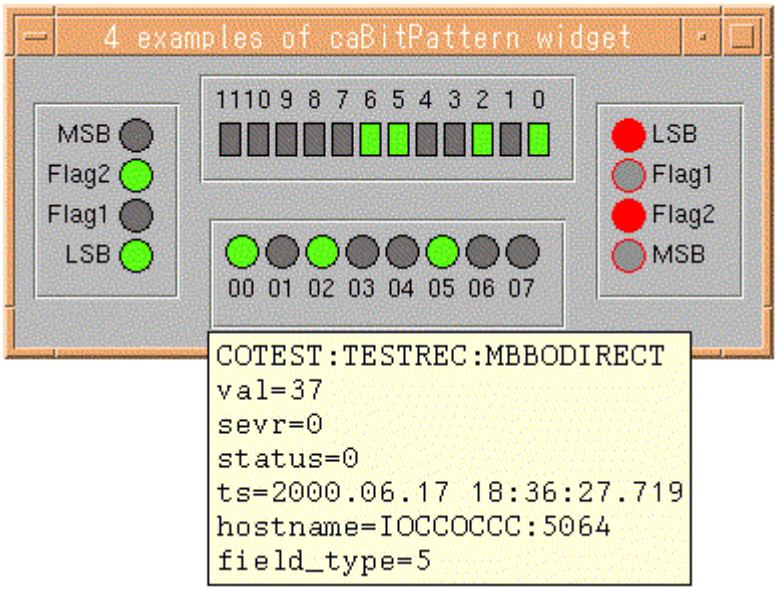

Figure 1: Four examples of caBitPattern widget in a top-level window. The middle lower one shows its CID (Channel Information Display) just below it. The right one is caused red by the MAJOR severity of the assigned channel.

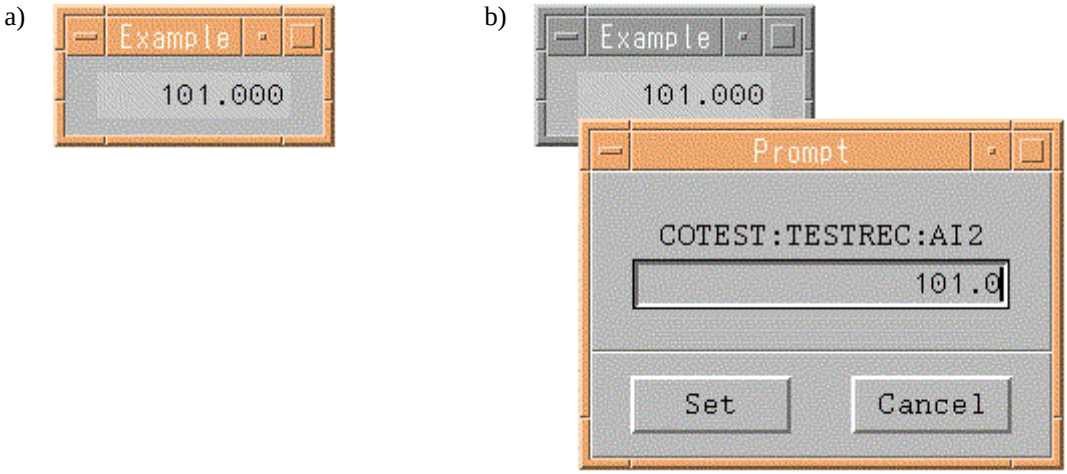

Figure 2: a) Example of a caWritableLabel widget. b) A dialog box appears to let user input a new value when the left mouse button is double-clicked on the caWritableLabel widget.

5 SUMMARY

Framework of the development of the Python/Tk widgets for EPICS based control system has been designed and implemented. Since all of the EPICS specific features are confined in caVar class and caWidget class, the design of a new widget gets much free hand. The object-oriented feature of Python allows even most of the already existing widgets to be customized easily for EPICS in a few lines of coding.

REFERENCES

[1] N. Yamamoto et al., “KEKB control system: the present and the future”, PAC-99, New York, 29 Mar.-2 Apr. 1999