Python による加速器制御用 GUI プログラミング

GUI PROGRAMMING FOR THE ACCELERATOR CONTROL SYSTEM USING PYTHON

中村達郎#, A), 青山知寛 B), 藤田誠 B), 中村卓也 B), 吉井兼冶 B)
Tatsuro Nakamura #, A), Tomohiro Aoyama B), Makoto Fujita B), Takuya Nakamura B), Kenzi Yoshii B)
A) High Energy Accelerator Research Organization (KEK)
B) Mitsubishi Electric System & Service Co., Ltd.

Abstract

KEKB and SuperKEKB accelerator control system has been built based on EPICS, which is the software framework for the distributed control system. Among the EPICS standard toolkits, there are several kinds of graphical editors to build the application programs with GUI (Graphical User Interface), such as MEDM, EDM and CSS BOY. Although they are powerful and easy to use without programming, another way to build applications using scripting languages such as Python is still available. The GUI programming is the more flexible way to develop applications but rather complicated. We have developed some programming tools in Python6 for the EPICS-based control system in order to make the GUI programming easy and efficient. These tools are presented in this paper.

1. はじめに

加速器の遠隔制御システムでは、中央制御室から操作を行なうためにグラフィカルユーザインターフェース（GUI）を備えた計算機プログラムを使うことが広く行われている。このようなプログラムはしばしば「操作パネル」と呼称される。GUI を使うことで加速器の運転員は直観的な操作が可能になる。KEKB/SuperKEKB 加速器では、制御システムを構築するためのソフトウェア・ツールキットとして EPICS (Experimental Physics and Industrial Control System) [1]を採用している。EPICS においても操作パネルを作成するためのツールが各種提供されている。特にグラフィカル・エディタを備え、操作パネルの設計・製作も GUI によって直観的に行なえるようなツールは非常に重要で、EPICS では Display Manager と呼ばれており、EDD/DM, MEDM, DM2K, EDM, CSS BOY など、幾多のツールが開発・提供されて来た。いずれもプログラミングを行なう事無しに操作パネルを作成することができ、計算機に習熟していないユーザでも素早い開発が可能で、効率よく開発が行える。

一方これと異なるアプローチとして Python などのスクリプト言語を用い、プログラミングによって GUI を構築して操作パネルを開発する方法がある。Display Manager では開発が容易な反面、GUI 部品（widget）は用意されているものから選ぶため機能や外見が制約されることになる。（Display Manager によってはユーザによる機能拡張がある程度可能なものもある。）また多数の制御点に対応して多数の widget を並べた操作パネルを作成するような場合、手作業で widget を配置して行くのは却って手間がかかり誤りも起きやすいため、作成の自動化を図る仕組みが必要になる。プログラミングによる方法では、これら制約のある Display Manager と比べて高度で柔軟な GUI を制限なく作成可能な反面、作成に手間がかかり、また技術の習得にも時間が掛かるのが難点であった。

KEKB 加速器では 2000 台を超える電磁石電源を扱うことから大規模な操作パネルの開発が必要であったこともあり、Python によるプログラミングによる操作パネルの開発が行われた。その過程でなるべく開発を容易にし、効率化するための様々な工夫が試みられた。以下では Python による開発環境について説明を加えた後、二つの試みについて報告する。

tatsuro.nakamura@kek.jp

2. Python による開発環境

Python は平易な文法で初心者でも学習が容易なインタプリタで動作するプログラミング言語である。平易なだけでなく、リスト、タプル、辞書といった高度なデータ構造を扱うことができ、オブジェクト指向プログラミングもサポートする。また各種の拡張モジュール（ライブラリ）の開発が盛んに行なわれており、様々な分野に適用することができる。GUI のための拡張モジュールはいくつかあるが、KEKB 加速器では Tcl/Tk とのインターフェース・モジュールである Tkinter を採用した。Tkinter は古くからあるモジュールで、基本的な widget は揃っており、複雑な GUI を構築するための機能も備えている。しかし複雑な制御用の GUI を作り込んで行くにはそれなりの量のプログラミングが必要である。

EPICS はネットワーク分散型の制御システムである。制御対象ハードウェアに接続された計算機（IOC, Input Output Controller）や、操作パネルなどの上位アプリケーションを実行する計算機（OPI, Operator Interface）など、多数の計算機が分散処理を行なうシステムに向いたソフトウェアであり、その中核となる通信プロトコルは Channel Access（CA）と呼ばれ、クライアント・サーバ・モデルに基づく。

CA で扱うデータ単位は制御点一点一点に対応した値（スカラー値、文字列、もしくは配列データ）であり、チャネル（Channel）と呼ばれる。チャネルはそれに付けられた名前で区別・参照される。CA の基本動作はある意味単純であり、値の読み取り（get）、値の書き込み（put）、値の監視（monitor）の 3 つがある。値の監視とは、値に変化があった時などに非同期的に通知が行われるもので、クライアント側では予め登録されたコールバックルーチンが呼び出される。CA のクライアント・ライブラリは様々なプログラミング言語での実装があり、Python では ca モジュールが作られている。

3. 制御用 widget ライブラリ

3.1 widget と EPICS の連携

一つ目のアプローチは、制御でよく使われる widget をライブラリとして用意しておくことで、操作パネル開発の効率化を図るものである[2]。さらに widget ライブラリそのものの開発も、EPICS CA と連携させるツールを用意することで定型化し、効率化を図った。

3.2 CA-variable

Tk では変数（variable）を定義することができ、ある種の widget では変数の値を表示したり、オペレータが入力した値を変数に受け取ったりすることができる。この便利なメカニズムを利用して、変数に CA のチャネルを結びつけることで、変数を通じてチャネルの値を表示したり、変数への入力値をチャネルに書き込んだりすることが簡単にできるようになる。Tkinter では変数は Variable クラスを継承して実装されている。そこで Variable クラスを継承し CA の機能を持たせた caVar クラスを作成した。このクラスから派生した 4 つのクラス（caStringVar, caIntVar, caDoubleVar, caBooleanVar）のインスタンスを CA-variable と呼ぶ。CA-variable には EPICS のチャネルと結び付けを行なう assign メソッドが用意されていて、チャネル名とモードを指定して呼び出す。モードには'monitor', 'put', 'both'があり、それぞれチャネルの読み出し、チャネルへの書き込み、読み書き両方を行なう事を指定する。

CA-variable を用いたプログラミングは、(1) CA-variable を作ってチャネルと結び付け、(2) それを Tk の一般的な widget に渡す、という手順になる。

```
#
# Example of customizing procedure
#   Label (Tk standard widget) --> caLabel (CA-widget)
#

import Tkinter
from caxtk import *

class caLabel(caWidget,Tkinter.Label) : # Base classes are caWidget and Label
    _widgetbase = Tkinter.Label         # Base widget class
    _mode = "monitor"                   # Mode of caVar
    _varkey = "textvariable"  # Option name to accept a variable in base class
    _ruleset = RULESET        # Predefined Configuration Rules

#
# Example of how to use a CA-widget (caLabel)
#

if __name__=="__main__" :
    root = Tkinter.Tk()         # create root window

    caLabel(root,channel="COTEST:TESTREC:AI",rule=1).pack()   # Simple example

    caLabel(root,channel="COTEST:TESTREC:AI",rule=1,          # More options
            conv=lambda x:x*1000,format="%10.3f mV",
            relief="groove",bd=2).pack(padx=8,pady=8)

    Tkinter.Button(root,text="Exit",command=root.quit).pack() # Exit button
    xca.pend_event(0.1)         # flush CA request
    root.mainloop()             # main event loop
```

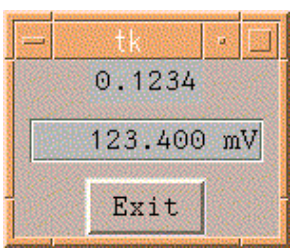

Figure 1: Sample program using CA-widget; the source code (left) and the result (right).

3.3 CA-variable のデータ変換と書式

CA-variable は GUI による入出力データと CA でやり取りされるデータを仲介するものと言えるが、その際にデータの加工を行なうこともできるようにした。conv オプションによりデータ変換関数を指定できる。また format オプションで書式を指定することでデータの文字列への変換を細かく指定することもできる。なお both モードでは読み書き両方を扱うため、conv オプションに加え、逆変換関数を rconv オプションで指定する。

3.4 CA-widget

CA-variable を使うことでプログラムが短縮化できるが、さらに一歩進めて CA-variable の生成と widget の生成を 1 度に行えるようにしたのが CA-widget である。これは一般の widget をベースに CA の機能を組み込み、EPICS 用にカスタマイズしたものであって、様々な widget があるがそれらを総称して CA-widget と呼んでいる。CA-widget を容易に開発するため caWidget という抽象クラスを導入した。CA-variable の生成や EPICS CA に関係する機能は caWidget クラスが集中的に受け持っている。ベースとなる widget と caWidget とを併せて継承することで簡単に CA-widget を創り出すことができる。ベースとなる widget は EPICS とは関係なく作られた汎用のもので良く、Tk の変数を受け入れるように作られていることが唯一の必須条件である。Figure 1 はプログラミング例である。ここの前半では caWidget クラスを用いて caLabel という名称で CA-widget のクラスを一つ定義し、後半では作成した caLabel クラスを使って簡単な表示パネルを作っている。

3.5 CID

さらに CA-widget には便利な機能を付加してある。その一つが CID（Channel Information Display）である。CID はチャネルの情報を表示する小さなウインドウで、CA-widget の上でマウスの中ボタンを押すと表示される。Figure 2 は CID の例である。CID の機能は caWidget クラスに実装しているため、caWidget を継承するだけで自動的に CA-widget に組み込まれる。

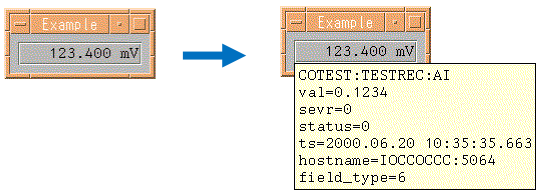

Figure 2: Example of CID.

3.6 DCR

もう一つの CA-widget の付加機能は DCR（Dynamic Configuration Rule）である。これはチャネルの状態に応じて色や外見を動的に変えて表示する仕組みである。典型的な例ではチャネルのアラーム状態に応じて黄色や赤色の警告表示にすることができる。Figure 3 は DCR で色を変えた見本である。色を変え方の規則は予め幾つか用意されていて CA-widget を使う時にユーザが簡単に選択することができる。また決められたものを選ぶだけでなく細かく指定してカスタマイズすることも可能である。DCR も CID と同様に、caWidget を継承するだけで自動的に CA-widget に組み込まれる。

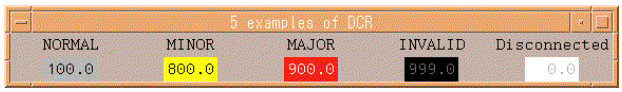

Figure 3: Example of DCR.

3.7 CA-widget の作成例

CA-widget として開発した例を Figure 4、Figure 5 に示す。Figure 1 では説明のために単純な例を示したが、これらの例は実際のプログラム開発に使っているものである。

Figure 4 は caBitPattern と呼ぶ widget で、データをビット列として扱い、各ビットのオン・オフ状態を一列に表示するものである。図では表示の向きや形態などが異なる 4 つの widget を並べている。また、CID の表示や DCR によって赤く警告表示がなされている様子も例示されている。

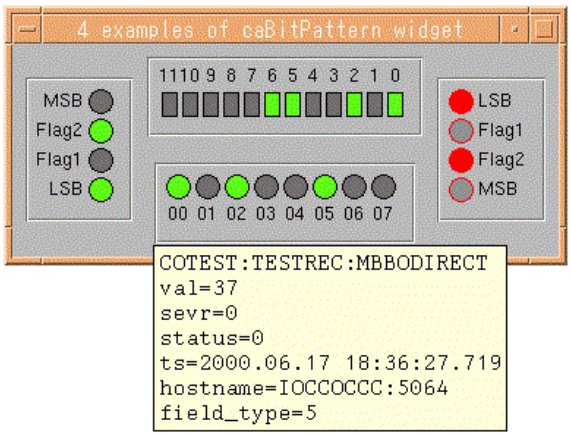

Figure 4: Example of caBitPattern.

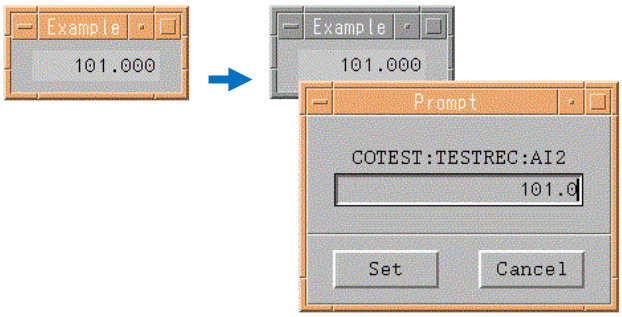

Figure 5: Example of caWritableLabel.

Figure 5 は caWritableLabel と呼ぶ widget で、文字列の表示を行なう Tk の Label widget を拡張したものである。通常はデータを文字列で表示しているだけだが、ダブルクリックすることで入力用のダイア

ログボックスが現れ、チャネルの値を書き換えることができる。

4. アプリケーション・ウインドウの雛形

二つ目のアプローチは、アプリケーション開発のベースとなるウインドウの雛形を用意することである。そこで試行的に簡単な雛形を作成してみた。Figure 6 は雛形を利用してアプリケーション・ウインドウを作った見本である。雛形は widget を配置するためのメインの作業領域を中心にして、メニューバー、コマンドボタン領域、ステータス表示領域を備えている。また、File メニュー、Help メニューを予め用意しており、画面コピーボタンも組み込まれている。また図では見えないが、バルーンヘルプを表示する機能も備えている。図の利用例では雛形を基に「Welcome !」という表示、「Misc.」メニュー、「45%」と書かれたボタンを追加している。

5. まとめ

プログラミング無しで操作パネルを作成できる Display Manager は強力なツールであるが、場合によってはプログラミングによる方法で開発することが適していることもある。プログラミングの柔軟性をなるべく損なわずに効率化を図るため、制御用の widget ライブラリを開発したり、アプリケーション・ウインドウの雛形を作成したりするなどの工夫を試みて来た。開発に使用した Python は学習が容易なスクリプト言語であり、プログラミングの専門家ではない制御システムのユーザにも使い易い。また Python ではオブジェクト指向プログラミングが可能なため、制御に必要な機能を組み込むといった、機能拡張の開発を行なうのにも適している。

一方、今回の試みでは Tkinter を採用したことで Tk に強く依存したものとなっている。GUI の構築のための拡張モジュールはいろいろな選択肢があるが、相互に互換性が無く共用も難しいため、どれを選択するかは悩ましいところである。一度採用したものから途中で乗り換えるのはコストがかかる。もっともこれは Python に限らず GUI のプログラミング全般に言えることであり、解決の難しい課題と言える。

今後は実際の加速器の制御・運転に即した操作パネルの使われ方を研究し、効率的な運転・開発をサポートできるようなツールを充実させて行きたい。

参考文献

[1] http://www.aps.anl.gov/epics/
[2] T. T. Nakamura, et al., “Development of the Python/Tk Widgets for the Control System based on EPICS”, EPAC 2000, Vienna, June 2000, p. 1865.

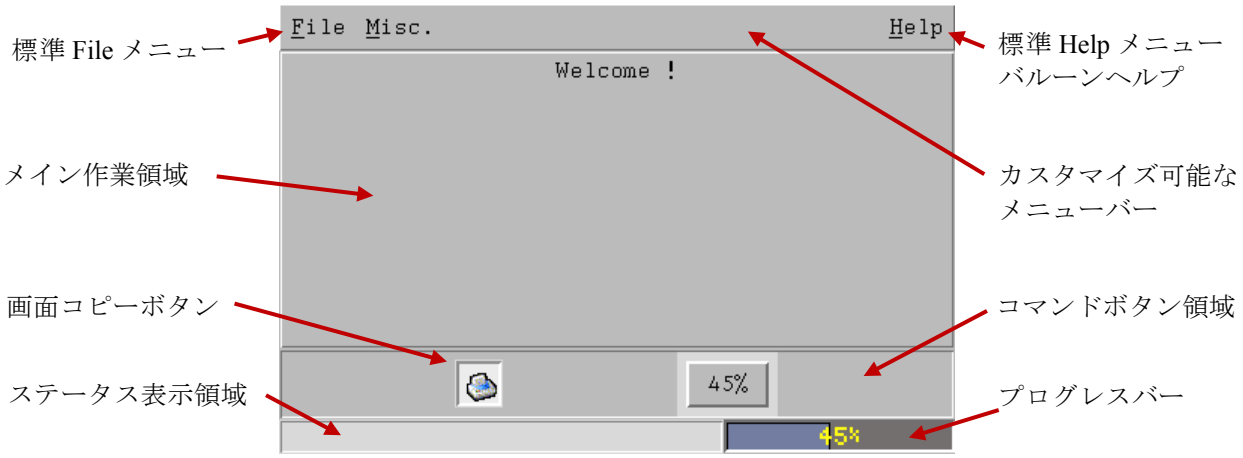

Figure 6: Template of the Application Window.