Proceedings of the 1st Annual Meeting of Particle Accelerator Society of Japan
and the 29th Linear Accelerator Meeting in Japan (August 4 - 6, 2004, Funabashi Japan)

RAPID APPLICATION DEVELOPMENT BY KEKB ACCELERATOR OPERATORS USING EPICS/PYTHON

M.Tanaka[1,A)], Y.Satoh [A)], T.Kitabayashi [A)], H.Iida [A)], T.Kawasumi [A)], K.Yoshii [A)],
T.Aoyama [A)], S.Shimomura [A)], K.Sugino [A)], T.Nakamura [A)], T.Ohkubo [A)], S.Fuke [A)], N. Yamamoto [B)]
[A)]Mitsubishi Electric System & Service Co. Ltd.
2-8-8 Umezono, Tsukuba, Ibaraki 305-0045, Japan
[B)]High Energy Accelerator Research Organization (KEK)
1-1, Oho ,Tsukuba, Ibaraki 305-0801, Japan

Abstract

In the KEKB accelerator facility, the control system is constructed based on the framework of EPICS. By using EPICS/Python API, which is originated from KEK, we can develop an EPICS channel access application besed on simple Python techology with only a few knowledge of EPICS channel access protocols. The operator's new tuning ideas are quickly implemented to the control system. In this paper, we introduce the EPICS/Python API and report the effectiveness of rapid application development by the KEKB operators using the API.

KEKB加速器運転員のEPICS/PythonによるRapid Application Development

１．はじめに

我々オペレーターは1999年からKEKBコミッショニンググループ（KCG）と共にKEKB加速器の運転を行なっている。KCGの主な役割はビーム調整や装置開発である。オペレーターはKCGの指示の下Ringへのビーム入射、ルミノシティ調整及び調整方法の改善、加速器の状態監視や安全管理等を行なっている。加速器を常時運転していると様々な監視対象や新しい調整アイデアが出てくる。これまではその度にKCG又は制御グループにモニター用のアプリケーション開発を依頼するか制御室内に点在する専用端末で監視していた。しかしアプリケーション完成までの時間がかかり過ぎたり、完成したアプリケーションに対する要求がいろいろあったりと問題点も多かった。オペレーターはソフトウェア技術の専門家ではなかったがKEKで開発されたEPICS/Python APIを使用することでオペレーターにもアプリケーションの開発が可能となった。オペレーターのアプリケーション開発への参加により上記の問題も軽減された。

２．KEKB制御システム

KEKB加速器の制御システムはEPICS[2]をベースに構築されている。EPICSとは加速器や大型測定装置、大型望遠鏡などの分散型制御システムを開発するためのソフトウェア開発環境と汎用アプリケーションのセットである。KEKBの場合、数台のUnix計算機によるオペレーターインターフェース部と約100台のIOC（デバイス制御用のVME計算機）による機器制御部の構成になっている。全IOC上のEPICSレコード数は約28万個ありこのレコードの状態の変化を知ることでオペレーターは加速器の状態を把握している。KEKではKEKBとPF-ARにEPICSを採用しており今後PFの一部でも採用が予定されている。

３．GUI開発環境

KEKBではGUI（Graphical User Interface）をMEDM、SAD/Tkinter、Python/Tkinterで開発しておりこれらを通して加速器の制御を行っている。

3.1 MEDM (Motif Editor and Display Manager)：EPICS標準のGUI作成ツールで簡単にパネル作成が可能である。しかし条件分岐などが出来ない為単純な操作や監視向きである。

3.2 SAD/Tkinter：KEKで開発された加速器設計の計算言語であるSAD[3]（Strategic Accelerator Design）にEPICSアクセス部分とTkのグラフィック表示を加えた言語である。KEKB加速器の制御用に使用されているアプリケーションはKCGの大半がSAD/TkinterのユーザーであることからSAD/Tkinterで作成されたものがほとんどを占めている。

3.3 Python/Tkinter：Pythonは比較的覚えやすく迅速なアプリケーション開発に適したオブジェクト指向型のインタプリタ型言語である。また構文がシン

[1] E-mail: tmanabu@post.kek.jp
[2] http://www.aps.anl.gov/epics/index.php
[3] http://acc-physics.kek.jp/SAD/sad.html

プルで便利な機能が多く組み込まれており、起動しただけで使用できる多数のコマンドが用意されている。そのほかにも豊富な拡張モジュールが提供されていて、プログラム開発者は拡張モジュールをimport(取り込み)するだけでその機能が使用出来る。

４．EPICS/Python API

EPICS/Python APIによってEPICSの詳細を知らなくてもPython言語を用いて制御パネルの開発が可能である。図1、2にEPICS/Python APIの概略図とソースコードの例を示す。

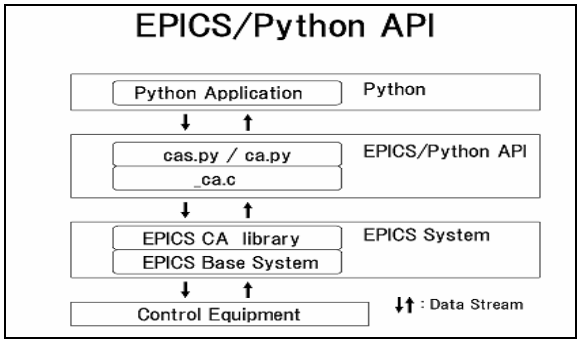

図1 EPICS/Python API 概略図

Python(Python Application)から送られた命令はEPICS/Python API （cas.py→ca.py→_ca.c）を経由してEPICS System（EPICS CA Library→EPICS Base System）からContorol Equipmentへと伝えられる。

```
from Tkinter import *
from cas import *          ← import文

class EPICS_CA:
    def __init__(self,master):
        frame=Frame(master)
        frame.pack()
        self.monival=Label(frame)
        self.monival.pack()
        print caget("COTEST:TESTREC:AI")
        caput("COTEST:TESTREC:AI",3)
        camonitor("COTEST:TESTREC:AI",self.monitor)
    def monitor(self,val):
        self.monival.configure(text=val[0])
root = Tk()
epics_ca=EPICS_CA(root)
root.mainloop()
```

図2 ソースコード

プログラム開発者はcas.pyをimportする（プログラムの最初に“from cas import＊”と書く）ことでEPICS/Python APIを利用することが出来る。プログラム中ではレコード名（チャンネル名）で指定されるEPICSレコードに対してCamonitor、Caget、Caputの３つのコマンドを使用すれば加速器に対する全ての制御を実現可能である。表1にコマンドの役割を示す。

表1 コマンドの役割

Camonitor	常にモニターし状態に変化があれば状態を返す
Caget	レコード状態を返す
Caput	レコードに対して値をセットする

５．EPICS/Pythonによる Rapid Application Development (RAD)

特定のレコード数個をモニターするパネルなどは形にこだわらなければ開発時間は1時間以内である。また他のモニターパネルを開発する時はモニターするレコード名を変えるだけでよいので数分で開発可能である。さらに条件文を追加することにより運転員が状況を把握しやすいようなパネルの開発も容易である。アイデアや要求があればすぐに追加や新規開発したりして数分後にはアプリケーションとしての使用が可能である。

オペレーターが開発したアプリケーションはKEKBのプログラムランチャーに登録されているだけで開発中を含め約90個ある。そのほとんどが運転を効率的に行うために開発されたもので、表2に運転開始から開発されたアプリケーション数を示す。2003/4からは飛躍的にアプリケーション数が増加しているがこれはEPICS/Python APIの有効性がオペレーターに広まったのと2003年度にソフトウェア技術習得機会を設けたことによると思われる。

表2 アプリケーション数

年月	アプリケーション数
2003/3 （技術講習会開催前）	約 10
2003/4～2004/6	約 80（SAD 20)

EPICS/Python APIを使った簡単なアプリケーションを図4、5、6に示す。

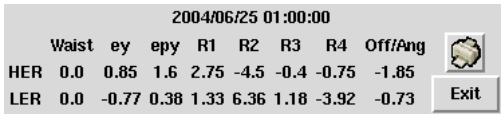

図4 ルミノシティ調整用Knobモニター

ルミノシティ調整用Knobのレコードをcamonitorすることで値が変化してもリアルタイムで値を把握することが出来る。

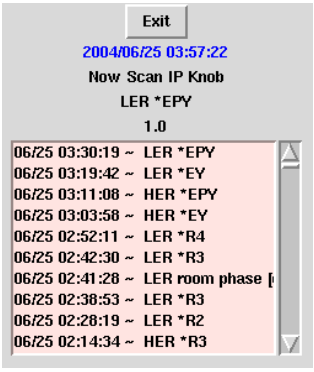

図5 ルミノシティ調整Knobの調整履歴モニター

ルミノシティ調整に使用中のKnob名とその値を表示し、値に変化があれば履歴リストに順に追加する。

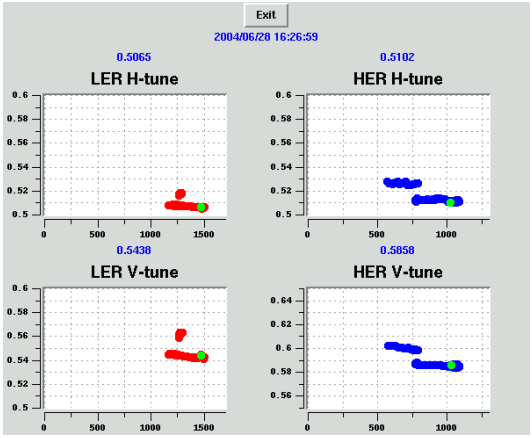

図6 ベータトロンチューンの履歴グラフ

ベータトロンチューンの値だけでも把握出来るが電流値と一緒にPython/Tkinterの拡張モジュールであるグラフ表示に追加することによって一目で状況が把握出来る。

このようなパネル以外にもEPICS/Python APIを利用したアプリケーションが多々あり[1] [2]加速器の運転状態把握に役立っている。

EPICS/PythonによるRADを通じてPython言語を習得することはSAD/Tkinterを使用したアプリケーション開発にも有効であった。SAD/TkinterにもEPICSアクセス機能がありEPICS/PythonによるRADと同様の事が可能である。図7にSAD/Tkinterで開発したルミノシティ調整用モニターパネルを示す。オペレーターのルミノシティ調整時に浮かんだアイデアや意見が集められ、それをオペレーターが開発に生かしている。パネルの使用者と開発者がオペレーターであることはパネルの開発に有効でありオペレーターのアイデアや意見が開発に素早く反映されることになる。

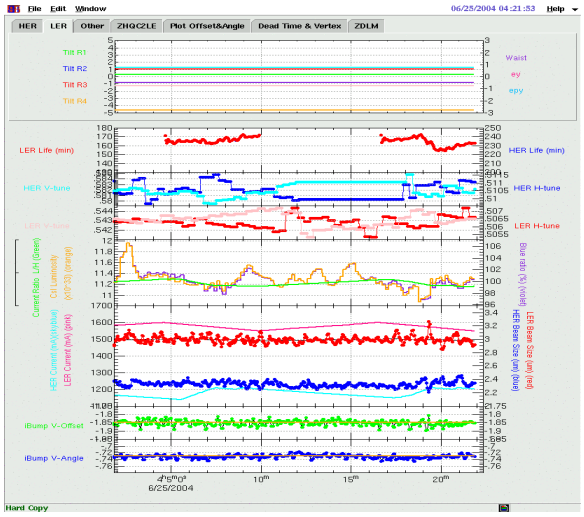

図7 ルミノシティ調整用モニターパネル

６．まとめ

プログラム経験の少ないオペレーターがアプリケーションを開発するのにEPICS/Pythonは有効であり、EPICS/Pythonを習得することによって得た知識はSAD/Tkinterを使用したアプリケーション開発にも有効であった。オペレーターがEPICS/Pythonを使って迅速にアプリケーション開発をすることによって加速器の運転状態を素早く把握することが可能になった。

謝辞

本論文を書くにあたりご助言、ご指導を頂きましたKEKBコミッショニンググループの方々にお礼申し上げます。

参考文献

[1] K.Yoshii, et al., “The Operator-developed Useful Tools at KEKB Accelerator”, Proceedings of Workshop on Accelerator Operation 2003, March 10-14, 2003

[2] K.Sugino, et al., “Effectiveness of Operation Tools Developed by KEKB Operatos”, 第１回日本加速器学会会年会・第29回リニアック技術研究会プロシーディングス, Aug. 4-6, 2004