Some Utilities for EPICS toolkit

T. KATOH, A. AKIYAMA, S. ARAKI, J-I. ODAGIRI, T. KAWAMOTO,
I. KOMADA, K. KUDO, T. NAITO,T. T. NAKAMURA, N.YAMAMOTO,
Accelerator Laboratory, KEK, Tsukuba, Ibaraki, 305 JAPAN
M. KAJI,
Mitsubishi Electric Co. Ltd., Japan
T. KITABAYASHI,
Mitsubishi Electric System & Service Engineering Co. Ltd., Tsukuba, Ibaraki, 305 Japan
N. KOIZUMI,
Mitsubishi Space Software Co., LTD.
M. TAKAGI, S. YOSHIDA,
Kanto Information Service, Tsuchiura, Ibaraki, 300 JAPAN

Abstract

KEKB control system is being build upon the EPICS(Experimental Physics and Industrial Control System) software toolkit. Although, it provides a rich set of tools to construct control systems, there is a room for further improvement. We have developed some tools to enhance usability of EPICS in KEKB control system. Outline of these tools will be reported.

1 INTRODUCTION

KEKB[1] is a project to build an asymmetric electron-positron collider in Japan. The first commissioning of the machine is scheduled in the fall of 1998. Double ring collider includes about 2,500 magnets. We expect more than 50,000 control points. Basic design of KEKB control system have been established in 1995[2]. Figure 1 shows a schematic view of the KEKB accelerator control system. The system consists of three layers,

- Presentation layer: A Unix server and X-terminals in the central control room,
- Equipment control layer:VME single-board computers distributed in the local control rooms, and
- Device interface layer:VME/CAMAC/VXI modules, GP-IB/ARCNET devices, Modbus+ modules, and devices communicated with RS-232.

Network equipments, server computers and some of VME computers have been installed and in operation[3]. This system is used for the development of applications and tools for KEKB control system. A part of KEKB beam transport line was commissioned using this system in the spring,1998.

We use the EPICS (Experimental Physics and Industrial Control System) [4] tool kit as the base of the KEKB control system. EPICS software includes,

- Driver software for the device interface layer,
- Distributed runtime database in equipment control layer,
- CA(Channel Access) protocol to exchange data between equipment control layer and presentation layer
- OPI tools(or CA client applications) for presentation layer,
- Utilities for configuration of these tools.

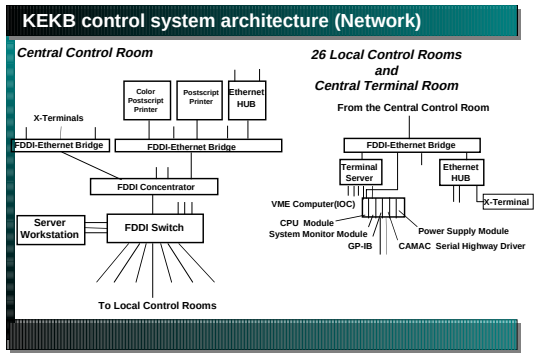

Figure 1: Schematic of the KEKB control system.

Although it is possible to build a full control system using just standard EPICS tools, there is a room for improvement.

For the development of KEKB control system, we have developed several tools for EPICS environment. In this article, two of them, “GDL”(GPIB device Description Language) and “ar2cwn” are described.

2 GDL/GDC: GP-IB DEVICE DESCRIPTION LANGUAGE AND ITS COMPILER

2.1 GP-IB device support in EPICS

Figure 2 shows a structure of EPICS software on VME computer. For a GPIB device support, EPICS supplies driver programs for GP-IB interface module and universal

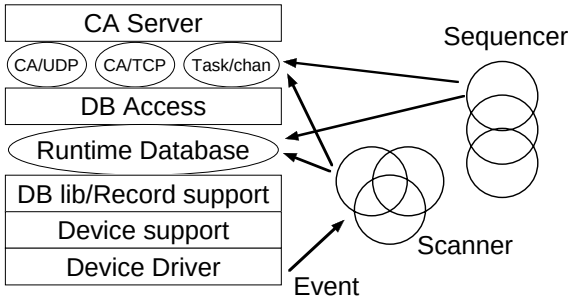

Figure 2: EPICS core softwares running on an IOC.

GP-IB device support routines(devCommonGpib.c). Providing a GP-IB command table, which should be written as an array in C language, for each GP-IB device, user can control GP-IB devices without coding of software, except when the device requires SRQ(service request) handler or other special data conversion routine. In the latter case, user have to supply these routines, also in C language.

Since this GP-IB command table is prepared as an array in C language. User must fill out every field in the table with the appropriate value. Some fields in the table have several meanings depending on the value of another filed. This diminishes readability and maintainability of the GP-IB command table.

GDL was designed to address the problem of readability and maintainability of the GPIB command table in EPICS GPIB support. In GDL, User can specify the value of field with the keyword. GDC supplies appropriate default values for most of fields if user does not specify it in the GDL file. These functionality reduces amount of coding to use GPIB device in EPICS and increase readability and maintainability.

GDL Format

```
Device DeviceName
NameTable NameTable_name=
name_1:val_1[,name_2:val_2...];
EfastTable Efast_name=name_1
[,name_2,....];
SrqHandler Handler_name param_name;
TimeOut Time;
%%in line C code
%{
    C codes
}%
ParamTable {
    param_name1 {
        rec = record_type,
    command="command string",
        [type = {read|write|readw|cmd|
        cntl|efasto|efasti|soft},]
        [pri=priority,]
        [rsp=buffer_size,]
        [leng=data_buffer_size,
        ]
        [conv=data_conversion
        _function]
                ...
            }
        [param_name2 {
                rec = bi,
                type = read,
                ...
                    } ... ]
}

%%in line C code
%{
    C code
}%
```

Figure 3: the GDL file format. Items in square brackets and C codes parts are optional.

Figure 3 shows general format of GDL file. *Device* statement defines a name of GP-IB device which this GDL file describes. Each entry in *ParamTable* should include at least an EPICS record type name, a GP-IB command string and

```
            Device(Tek2430)
          NameTable of-
fOn = "OFF", "ON";
          ParamTable{
            set_ch1_volt {
              rec=ai,
              command="CH1? VOL\r",
              conv="CH1 VOLTS:%lf"
            }
          }
```

Figure 4: SAMPLE GDL file. GDC output from this file is shown in Figure 5 on the next page.

GDC output SAMPLE

```
#define DSET_AI     devAiTek2430Gpib
gDset DSET_AI   = {6, {report, init_dev_sup, devGpibLib_initAi, NULL,
devGpibLib_readAi, NULL,
    (DRVSUPFUN)&devSupParms,(DRVSUPFUN)devGpibLib_aiGpibWork,
(DRVSUPFUN)devGpibLib_aiGpibSrq}};
static char             *offOnList[] = { "OFF", "ON" };
static struct devGpibNames offOn = { 2, offOnList, NULL, 1 };
static struct gpibCmd gpibCmds[] = {
    /* CMMAND 0 set_ch1_volt */
  {&DSET_AI, GPIBREAD, IB_Q_LOW, "CH1? VOL\r", "CH1 VOLTS:%lf", 0, 32, NULL, 0, 0,
NULL, NULL, -1 }
};
static struct  devGpibParmBlock devSupParms = {
  &Tek2430Debug,         /* debugging flag pointer */
  -1,                   /* device does not respond to writes */
  TIME_WINDOW,          /* # of clock ticks to skip after a device times out */
  NULL,                 /* hwpvt list head */
  gpibCmds,             /* GPIB command array */
  NUMPARAMS,            /* number of supported parameters */
  -1,                   /* magic SRQ param number (-1 if none) */
  "devTek2430Gpib",     /* device support module type name */
  DMA_TIME,             /* # of clock ticks to wait for DMA completions */
  NULL,                 /* SRQ handler function (NULL if none) */
  NULL                  /* secondary conversion routine (NULL if none) */
};
```

Figure 5: a sample of GDC output.

a name of the entry. GDC converts GDL description into C program as shown in Figure 5. It will be compiled in VxWorks/Tornado for downloading.

We used yacc and lex programs to implement GDL parser.

3 AR2CWN

EPICS standard archiving tools, AR and AR_cmd , can store data in ether binary or ascii format. EPICS retrieval tool(ARR/ARR_cmd) can read both formats and present data in textual or graphical way. Unfortunately ARR/ARR_cmd cannot handle wave form data(array of data) properly. We took PAW(Physics Analysis Workstation)[5] developed at CERN for displaying and analyzing data collected by EPICS archiver. PAW has several advantages as a data analysis tool such as:

1. it has a powerful functionality to analyze data,
2. GUI version(PAW++) is available,
3. it is widely used in HEP community.

A cwn(column wide N-tuple) format is a standard data format used in PAW/PAW++. Although PAW/PAW++ can

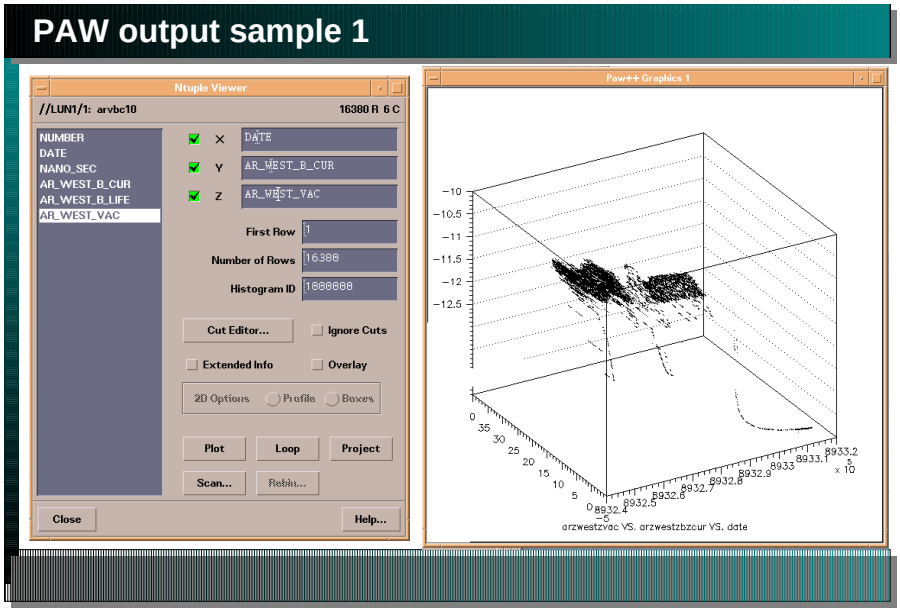

Figure 7: A sample plot produced by PAW++. User can configure a plot using graphical user interface shown left.

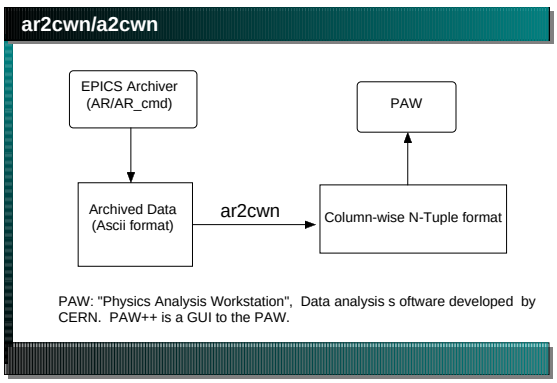

Figure 6: "ar2cwn" converts a data file stored by EPICS into column wide N-tuple format.

read other data format, cwn format is an efficient way to supply data to PAW/PAW++.

"ar2cwn" is a utility program which converts an ascii format file produced by AR/AR_cmd into cwn format file. After conversion, user can use PAW/PAW++ to analyze archived data. Macro capability in PAW can be used to manipulate stored data. Figure 7 shows a sample output from PAW++. The graph shows the relation between a beam current and logarithm of vacuum pressure in AR storage ring in KEK.

4 CONCLUSION

We have developed several utilities for EPICS tool kits. These small programs enhances usability of EPICS tools considerably. Although these tools are developed in KEKB project, they are general tools and can be used in any project based on EPICS tool kit.

5 REFERENCES

[1] "KEKB B-Factory Design Report", KEK Report 95-7, August 1995

[2] T.Katoh et al., "Status of the KEKB accelerator control system development's," in Proceedings of the 1995 International Conference on Accelerator and Large Experimental Physics Control Systems, M. Crowley-Milling, P. Lucas and P. Schoessow, Eds., pp 899 - 891

[3] T.Katoh et al., "Present status of the KEKB control system",ICALEPCS97, Beijing, China, 1997

[4] B. Dalesio et al. "Distributed Software Development in the EPICS Collaboration," in Proceedings of the 1995 International Conference on Accelerator and Large Experimental Physics Control Systems, M. Crowley-Milling, P. Lucas and P. Schoessow, Eds.,pp 360 - 366. ; "EPICS home page", http://epics.aps.anl.gov/asd/controls/epics/Epics-Documentation/WWWPages/

[5] "Physics Analysis Workstation - PAW ", http://wwwinfo.cern.ch/asd/paw/index.html