Integration of the Modeling Program with Accelerator Control Systems in TRISTAN

A. Akiyama, H.Fukuma, T. Kawamoto, S. Kamada, S. Kuroda, M. Kikuchi,
S. Matsumoto, T. Mimashi, K. Oide and N. Yamamoto
KEK, National Laboratory for High Energy Physics
1-1 Oho, Tsukuba, Ibaraki, 305 JAPAN

S. Yoshida
Kanto Information Service
8-21 Bunkyo, Tsuchiura, Ibaraki, 305 Japan

Abstract

SAD (Structured Accelerator Design) is a modeling program developed and widely used in KEK. The rejuvenated TRISTAN control system uses SAD as an integral part of the system. The main program of the rejuvenated control system, which has a graphical user interface based on X-window system on a UNIX work station, uses SAD for optics calculation, beam orbit correction and orbit bump calculation. Optics data and orbit data are stored in the format which can be directly read by SAD, so that accelerator physicists can use these data for off-line analysis using SAD. A beam-based alignment method and an automated luminosity optimization program have also been developed and tested in TRISTAN using the rejuvenated TRISTAN control system and SAD. Integration of SAD with the KEKB accelerator control system will be also discussed.

I. Rejuvenated TRISTAN accelerator control system

The main TRISTAN control system has been used for more than ten years [1]. It becomes difficult to maintain this system and to get parts for repair. It also became difficult to fulfill the increasing demands on the accelerator control system because of limited CPU power and of limited storage space in the main TRISTAN control system. The rejuvenation of the TRISTAN control system was planned as a solution [2].

The rejuvenated control system became available in 1993 and has gradually evolved since then. The rejuvenated TRISTAN control system uses UNIX workstations running X-windows as a operator interface, as shown in Figure 1. The main program sends commands to or receives data from a VME single board computer running OS9 through TCP/IP socket communication. The VME computer writes a command into CAMAC memory module through the CAMAC driver in a VME sub-rack and raises a LAM signal. A process in a HIDIC mini-computer in the main TRISTAN control system detects the LAM and starts appropriate processes to finish the task. To return the result to the workstations, the HIDIC mini-computer writes data into the memory module and sets flags at the specified area. A process on the VME checks this area periodically. When it detects changes of the flag, it reads data from the memory module and send them back to the UNIX workstations through a socket connection.

Figure 2 shows a main panel of the control system on Unix workstation used as a operator console. Tasks covered by this panel are:

1. read/write/save/restore/display/calculate optics.
2. read/write/save/restore/display/calculate correction magnet settings.
3. measure/save/restore/display/calculate beam orbit.

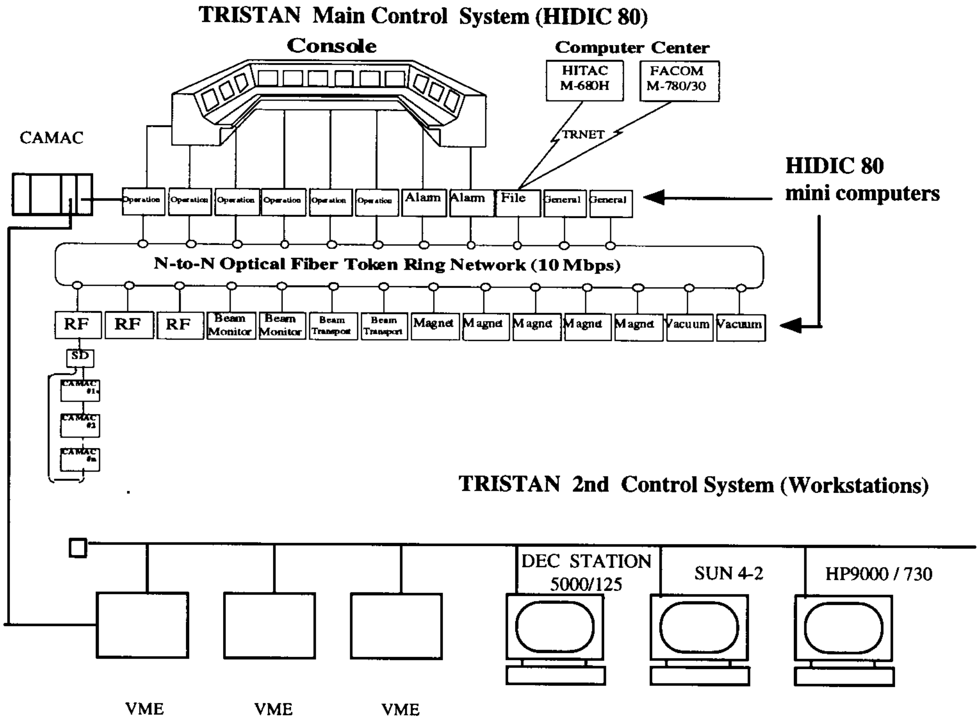

Figure. 1. System architecture of the rejuvenated TRISTAN control system.

The use of UNIX workstations allows us to store a large amount of data into the Unix file system. Unix workstations also provide a good application development environment.

II. SAD: Structured Accelerator Design

The rejuvenated control system uses SAD/FFS (Structured Accelerator Design/Final Focus Subsystem), the accelerator modeling program developed at KEK since 1989, wherever an accelerator model is required in the control system. The use of a single modeling code in the control system simplifies the management of data which describe an accelerator model.

The main coding language of SAD is Fortran. C is also used to access system functions. The SAD program includes more than 85,000 lines of code. The current version of SAD runs on DEC/Alpha Workstation/Servers running Digital Unix (formally known as OSF1) and on Hewlett Packard HP9000s 700 series workstations running HP-UX 9.05. Both system can compile the same Fortran code of SAD.

The SAD top-level routines define the input data format to describe the accelerator model. It is a dialect of the standard accelerator input format [3]. Figure 3 shows a sample of the input data. The SAD top-level stores data as internal data structures in a dynamically-allocated memory space.

After reading all the model description data into the top-level routines, control of the program is passed to FFS. FFS reads a user command and returns the result of the command. Command syntax is similar to that of Mathematica [4]. SAD/FFS supports a double precision number, a character string, a list structure and a pure function as primitive data types.

As an optics calculator, SAD handles full four-dimensional phase space. It can calculate projected and diagonalized Twiss

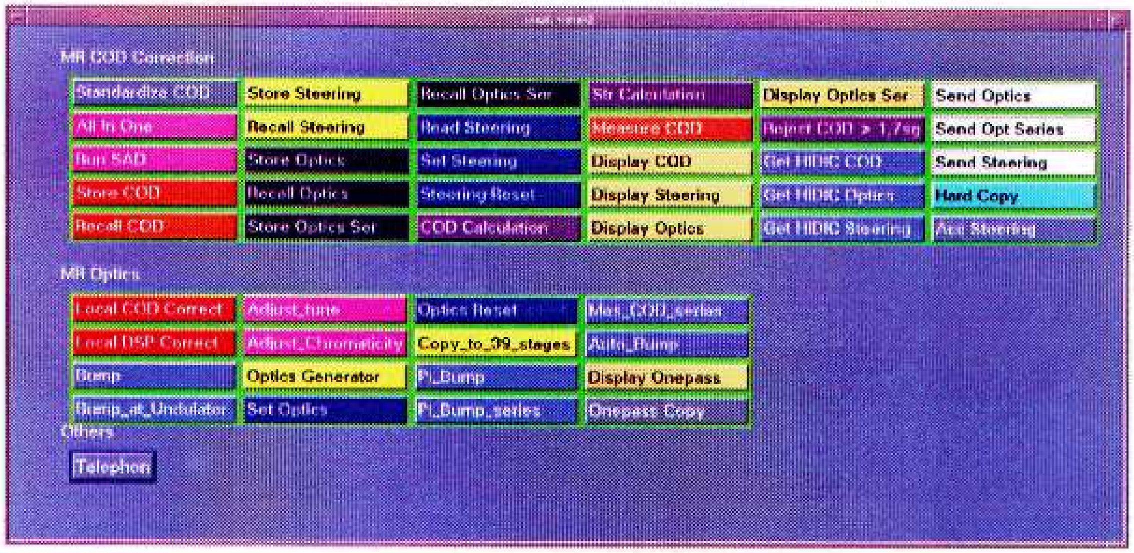

Figure. 2. Main panel of the rejuvenated TRISTAN control system.

```
 DRIFT  L1      =(L = 4.6443133730064 )    L2      =(L = 0.6391684124701 ) ;
 BEND   B       =(L = 5.9078   ANGLE = .055742725957    E1 = .5    E2 = .5 ) ;
 QUAD   QD1     =(L = 0.503     K1 = -0.1181376408391 )
        QF1     =(L = 0.818     K1 = 0.1904715700307 )
        ;
 SEXT   SD1     =(L = 0.5   K2 = 0 )          SF1     =(L = 0.15 K2 = 0 );
 MARK   NMARK   =(AX = 0   BX = 8.0 AY =0     BY = 30.0
                  EX = -0.145    EPX=0    EMITX = 1.80E-8 EMITY = 1E-12
                  DP = 6.67E-4    AZ = 7.4E-4   DZ = 7.4E-4)
        ;
LINE
      ASC=(IRTN RFSYM RFSECT RFSYM IROT)
      ;
ffs use ASC;
```

Figure. 3. A sample of SAD input data

parameters. FFS has a powerful yet flexible optimizer to find the magnet setting which satisfies the matching conditions. As a tracking code, SAD tracks particles in full 6-dimension phase-space. SAD uses a symplectic integration algorithm for tracking particles.

Data used by the rejuvenated control system are saved in the format which can be read by SAD/FFS. It allows an accelerator physicist to use these data for off-line analysis without further programming. This feature allows accelerator studies which were not possible before. Studies on the long term stability of the beam orbit and on beam-based alignment of sextupole magnets are examples of the accelerator studies performed with this rejuvenated control system.

A. Long term stability of the beam orbit in the TRISTAN MR

A slow drift of orbit was observed in the TRISTAN Main Ring (MR). This orbit drift degrades luminosity considerably. This kind of orbit drift can be a serious problem in the KEK B-factory (KEKB) which has a tighter tolerance for magnet misalignment than the TRISTAN MR. A study of long term orbit stability was carried out to determine the rate of long term orbit drift.

Orbit data from the TRISTAN MR were measured every 3 minutes during one week of a study period and stored in files. More than 3000 data sets were taken and occupy more than 20 MB of disk space. Data was analyzed off-line using SAD. Figure 4 shows the rms position variation of the orbit as a function of time. Analysis suggests that the transverse movement of quadrupole magnets in the interaction region of the MR is the main source of the drift.

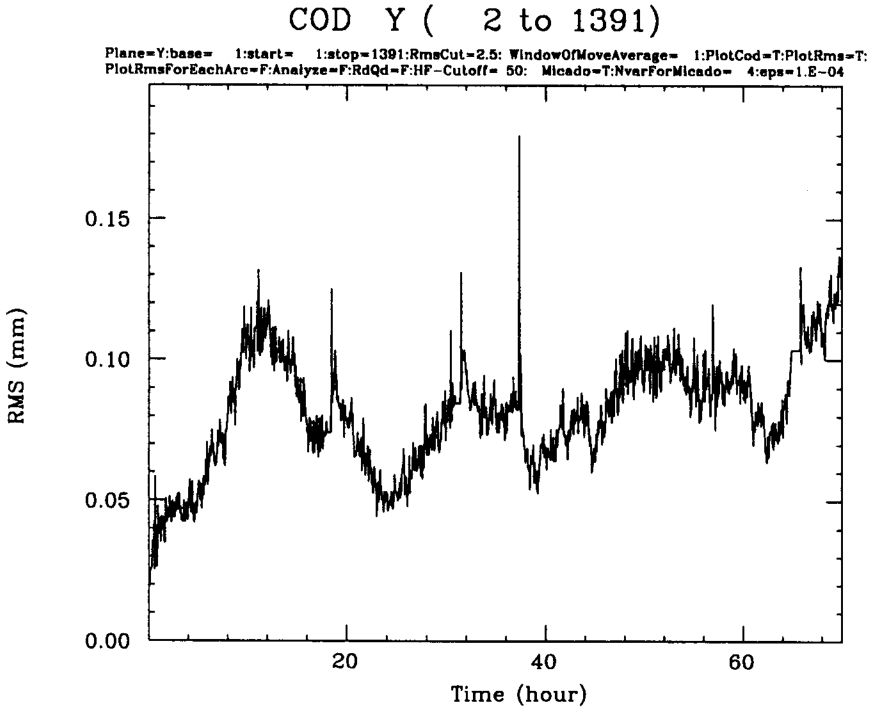

Figure. 4. RMS value of vertical orbit (mm) is shown as a function of time (hour). It shows slow drift of beam orbit.

B. Beam-based alignment of sextupole magnets

Sextupole magnets in a beam bump region can cause a perturbation of the orbit outside the bump region. Measurement of this perturbation as a function of bump height gives us information about the magnetic center of the sextupole magnet. This technique has been tested in the TRISTAN MR. A program written for this study reads a configuration file, sets a bump orbit, measures a beam orbit and stores it. Stored data are analyzed using SAD. Figure 5 shows a result of this analysis. The bottom of the parabola in the lower left graph corresponds to the magnetic center of a sextupole magnet.

III. SAD in KEKB accelerator control system

EPICS software tools [5] have been chosen as a basis for the KEKB accelerator control system [6], [7]. To integrate the SAD program into the EPICS-based KEKB control system, we are developing an interface to the EPICS run-time database in SAD. Current implementation supports six channel access functions, CaOpen/CaRead/CaWrite/CaOpen-Array/CaReadArray/CaWriteArray. CaOpen takes a character string as an argument and returns a channel id number. CaRead takes a channel id number as an argument and returns the current value of channel. CaWrite takes two arguments, a channel id number and a new value and returns the current value of the channel. CaOpenArray, CaReadArray and CaWriteArray are extensions of CaOpen/CaRead/CaWrite. These functions take lists as arguments.

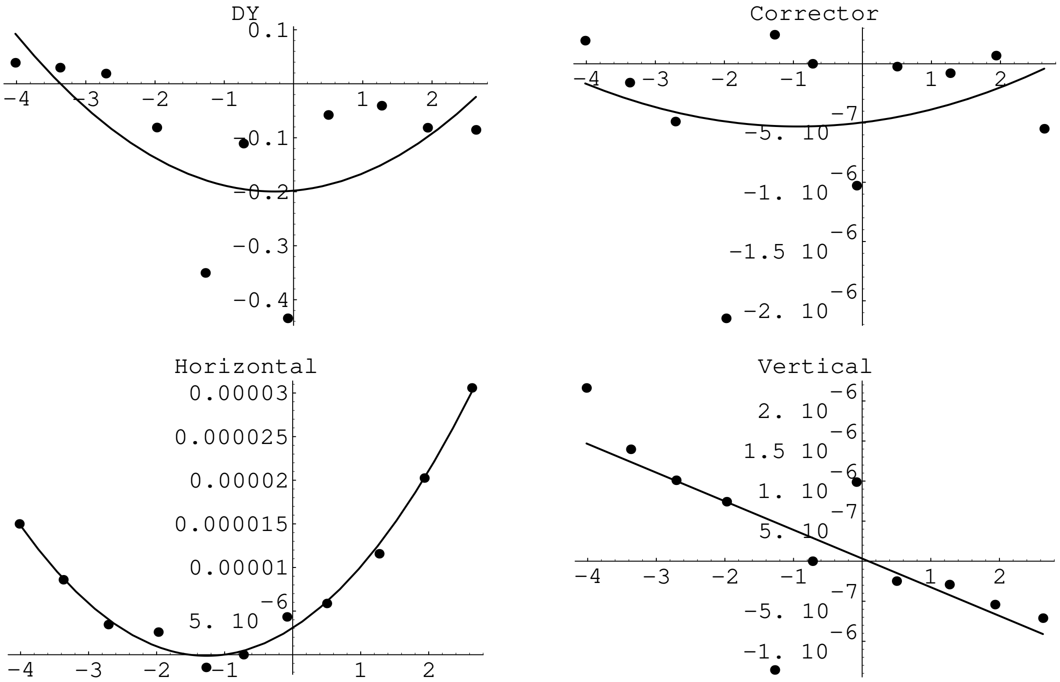

Figure. 5. The lower two graphs show the strength of correctors needed to compensate the perturbed orbit as a function of bump height at a sextupole magnet. Upper graphs indicate residual errors.

```
%./Casad1.exe
  cpu time= 3.0000E-02(sec) dt=     30.000(msec) free area::   40721
off ctime; mark m=();line l=(m); ffs use l;
 Design orbit length: .0

***/M:***>  names={"temperature:1","temperature:2","temperature:3"};

***/M:***>   chs=CaOpenArray[names]     !returns a list of Channel ids.
{1077998648.4999999,1078045344.5,1078045432.5}

***/M:***>   CaReadArray[chs]           !Read value of channels.
{1,2,3}
```

Figure. 6. Sample session of SAD with Channel Access interface

Having such a CA interface in SAD, a script shown in Figure 7 can be a part of an orbit correction program in the KEKB accelerator control system. It adds great flexibility to the control system.

IV. Conclusion

Experience with the rejuvenated TRISTAN accelerator control system shows the importance of the integration of a modeling program into an accelerator control system. The KEKB accelerator control system presents an opportunity to build such an integrated control system from scratch. The EPICS database design simplifies an interface between an EPICS-based control system and the modeling program. It opens the possibility of controlling a real accelerator from the modeling program.

```
read "LER.skeleton.sad";
ffs use LER;
steer=LINE["ST*","NAME"] ! it returns a list of steering names.
stchname=FindChanName[steer] ! get channel names from element name
used in SAD.
strchs=CaOpenArray[chname]
monitors=LINE[''M*'',NAME]
MonChnnelName=FindChanName[monitors]
monchs=CaOpenArray[MonChnnelName]
orbit=CaRead[monchs] !read  BPM values via EPICS
optics=CalculateOptics[]
CorrectOrbit[optics,"XY",steer,orbit] ! calculate optics
DisplayOrbit[optics,orbit]
If[AskOperatorResponse["Shall I set this steering?"]=="Yes",
CaWriteArray[steerchs,LINE[steer,"KICK"]]]
```

Figure. 7. Possible fragment of SAD scripts used in KEKB

References

[1] S-I Kurokawa, et al., “The TRISTAN Control System”, Nucl. Instr. and Meth., A247, (1986) pp. 29-36.

[2] T. Mimashi et al., “The rejuvenation status of TRISTAN accelerator control system”, Nucl. Instr. and Meth., A352 (1994) 128-130.

[3] D.C. Carey and F.C. Iselin, “A standard input language for particle beam and accelerator computer program”, CERN/SL/90-13

[4] S. Wolfram, “Mathematica: A system for Doing Mathematics by Computer”, Second Edition, Addison-Wesley Publishing Company, 1991

[5] L.R. Dalesio, et al., “EPICS Architecture”, ICALEPCS 91, KEK Proceedings 92-15, (1992) pp.278-282.; L. Dalesio et al. “The Experimental Physics and Industrial Control System Architecture,” ICALEPCS 93, Berlin,Germany, Oct. 18-22, 1993.

[6] S-I Kurokawa, “ Status of TRISTAN-II Project”, Proceedings of the 1993 Particle Accelerator Conf, pp. 2004-2006; S-I Kurokawa, “KEKB status and Plans”, 1995 Particle Accelerator Conference and International Conference on High-Energy Accelerators, Dallas, Texas,USA, May 1-5, 1995.

[7] T.Katoh et al., “Status of the KEKB accelerator control system development”, these Proceedings.