ACCELERATOR OPERATION WITH GENERAL-PURPOSE COMPUTER

E. Kikutani, A. Akiyama, T. Katoh, H. Koiso, S. Kurokawa and K. Oide

KEK, National Laboratory for High Energy Physics
Oho-machi, Tsukuba-gun, Ibaraki-ken, 305, Japan

Abstract

The TRISTAN Accumulation Ring is being operated under a control system which consists of a token-passing ring network of eleven minicomputers. Tasks that overload these minicomputers are processed on a back-end general-purpose computer, M-200H, which is one of the processors of the KEK central computer system. One minicomputer in the ring network and M-200H are connected via KEKNET, an in-house high-speed network at KEK. The software system of the TRISTAN control is based on the NODAL interpreter language. The communication with the back-end computer through KEKNET is done using network functions which are NODAL callable subroutines. Functions for reading/writing a disk data-set on M-200H, submitting a batch job, reading the status of the submitted job, etc. are prepared.

Introduction

TRISTAN[1] is an electron-positron collider facility, now being constructed at National Laboratory for High Energy Physics (KEK). It consists of three accelerators: 2.5 GeV Linac, 8 GeV Accumulation Ring (AR) and 30 GeV Main Ring (MR). The construction of AR has been completed in October 1983 and now its characteristics are being studied using electron beam.

AR and MR are controlled by a single system which consists of 24 minicomputers and a back-end computer. The minicomputers actually control the accelerators, while the back-end computer processes tasks which overload the minicomputers. As the back-end computer we adopted a general-purpose computer, HITACHI M-200H, which is one of the processors of the KEK central computer system.

We can identify some advantages of using a general-purpose computer for accelerator operation as a back-end computer. The first is continuity of works of designs and operations of accelerators. At KEK, and maybe at other institutes, lattice designs are carried out using a general-purpose computer. Many programs used at design steps are also useful for operation of the accelerator; therefore it is convenient to use a general-purpose computer also as a back-end computer.

The second advantage is that general-purpose computers have rich resources, such as high speed calculation power, large memory space, large disk space, various useful peripherals, etc. For example, M-200H which we are using as a back-end computer has a calculation speed of 4.56 Mega FLOPS and 12-Mbyte memory. A typical program used for an orbit correction (we will describe it later) necessitates 4-Mbyte memory and 10-second CPU time by M-200H. Then if we want to operate the accelerator without wasting long waiting time, it is necessary to use a computer of this class.

On the other hand, there are disadvantages of using a general-purpose computer. The typical one is the slow or unpredictable response time due to sharing the CPU and other resources with other batch jobs. We have overcome this difficulty by using a special job class that has a higher priority in job scheduling.

Hardware system

At present AR is operated under a computer control system consisting of eleven minicomputers (HITACHI HIDIC 80E's) that make a subset of 24 minicomputers mentioned above. These computers are linked together by a token-passing ring network of optical-fiber cables with the transmission speed of 10 Mbps. Figure 1 illustrates the structure of the computer control system.[2]

One minicomputer (LIBRARY) in the network and M-200H are connected with KEKNET.[3] KEKNET was originally designed for transferring data produced by high-energy physics experiments from data-taking minicomputers to the KEK central computer system. The maximum data transfer rate of KEKNET is 500 Kbyte/s.

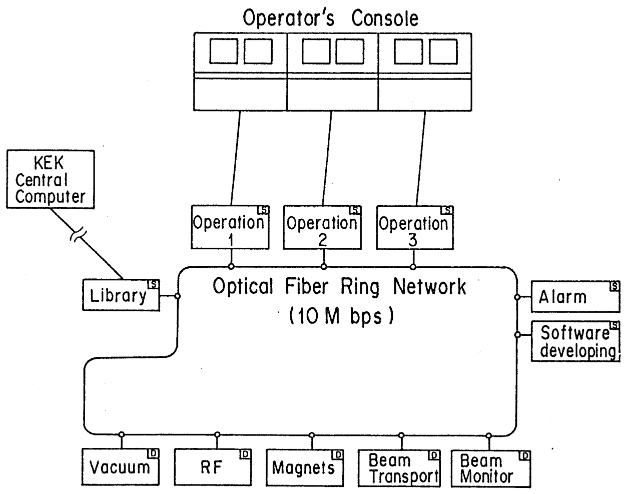

Fig. 1 TRISTAN Accelerator control system

Figure 2 explains the KEKNET system. Data are transferred through two repeater computers. The repeater computer I (HIDIC 80), which is located near M-200H, is connected to the channel of M-200H. The repeater computer II (HIDIC 80E) is located at the Counter Experiment Hall, where high-energy physics experiments are made. It communicates with data-taking computers of high-energy physics experiments and also with the LIBRARY computer of the TRISTAN control system. These two repeater computers are linked to each other by the same type ring network as the TRISTAN control system.

The back-end M-200H is one of the processors of the KEK central computer system. This system consists of three M-200H's named CPU A, CPU B, and CPU C, which are connected

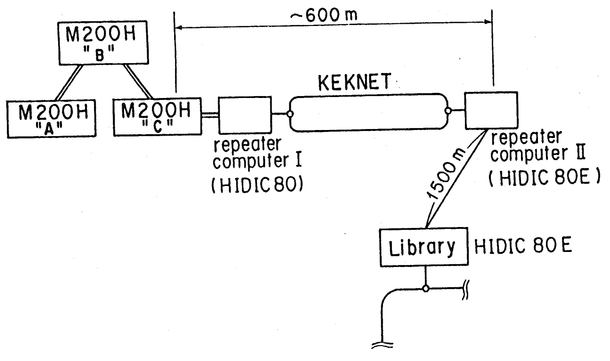
~600 m
M200H "B"
M200H "A"
M200H "C"
KEKNET
repeater computer I (HIDIC80)
repeater computer II (HIDIC 80E)
1500 m
Library
HIDIC 80E

Fig. 2 KEKNET and its related system

to form a loosely coupled multi-processor system. The CPU A with 12-Mbyte memory is used for long-time batch jobs, and the CPU B with 16-Mbyte memory is used mainly for TSS terminal jobs and partially for batch jobs. The CPU C with 12-Mbyte memory is used both for relatively short-time batch jobs and for on-line real-time processing jobs. The repeater computer I is connected to the CPU C. This three-CPU system is installed in the KEK central computer building; the distance from the TRISTAN control room is about 1 km.

Software system

The software system of the TRISTAN control[4] is based on the NODAL system. NODAL is an interpreter language and was originally devised at CERN for controlling SPS.[5] In our system all application programs are written in NODAL, while basic device handlers and some complicated routines are written in a compiler language PCL (Process Control Language prepared by HITACHI) as subroutines that are called from NODAL. Following the SPS NODAL system, we call device handlers data modules, and other subroutines NODAL functions. Examples of the NODAL functions are graphic CRT handlers and mathematical functions. The NODAL system dynamically links the interpreter and these subroutines at run time.

Table 1 Summary of network functions

FALOC	Allocate a data-set
FRARAY	Read data from a data-set and set them into a NODAL array
FWARAY	Write data stored in a NODAL array on a data-set
FRSTR	Read data from a data-set and set them into a NODAL string variable
FWSTR	Write data stored in a NODAL string variable on a data-set
FCLOSE	Close and free a data-set
FSMBIT	Submit a batch job
FSTATU	Read status of the submitted job

The communication with the back-end computer through KEKNET are done by network functions. Table 1 summarizes the basic network functions. For disk I/O action four functions FRARAY, FWARAY, FRSTR, and FWSTR are prepared. The former two functions transfer data to/from a NODAL array from/to a disk data-set on M-200H, while the latter two functions transfer data to/from a NODAL string variable from/to a disk data-set. In both cases disk I/O is done record by record. With the function FSBMIT, we can submit a batch job on M-200H; users can use the same load module and the same job control language (JCL) as the off-line analysis. This scheme guarantees the continuity between off-line tasks and on-line accelerator operation. The status of the submitted job can be known by the function FSTATU.

A typical sample program using these functions is as follows:

```
1.10 CALL FALOC ('R', 'TEST', FN, RL)
1.12 $SET ST=FRSTR(FN):3.10
1.14 TYPE ST !
1.16 GOTO 1.12

3.10 FCLOSE(FN)
3.12 END
```

This program reads data from a data-set named TEST on M-200H, which contains some EBCDIC code information, and types them on a terminal, record by record. The first line 1.10 allocates the data-set for reading (the argument 'R' denotes that) and the lines 1.12 through 1.16 read the records successively until the end of the data-set is detected. The line 3.10 closes and frees the allocated data-set. If one wants to allocate a data-set for writing, he must write,

```
1.10 CALL FALOC ('W', 'TEST', FN, RL)
```

instead of the line 1.10 of the example above.

For submitting jobs one can use the function FSBMIT:

```
CALL FSBMIT (DN, JN)
```

where DN is the name of the data-set which contains the JCL and JN is the job number given by the function. JN is used later to identify each submitted job in the function FSTATU. As we have seen, we can access to the back-end computer within the framework of the KEK NODAL system.

The software structure in the back-end computer is illustrated in Fig. 3. The On-line Control Program (OCP), which has been started just after initial program loading of M-200H, controls flows of data on KEKNET and supervises resources concerning the network. Under the control of OCP, the Accelerator Operation Monitor Program (AOMP) is running. Since OCP carries out actual I/O processes, AOMP and other on-line jobs communicate with OCP by calling some subroutines prepared in the system library. AOMP is running throughout the accelerator operating hours. It waits for the interruption which indicates that OCP has received the data from KEKNET. On interruption it will get data, analyse them, and take the necessary action. After completion of the action, AOMP sends the response and related data to KEKNET. If necessary AOMP converts character codes between ASCII and EBCDIC and representation format of floating

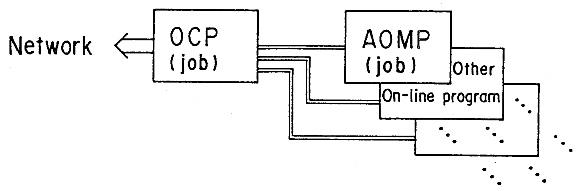

Fig. 3 Software structure in the back-end computer

point numbers.

As explained above, AOMP works like a terminal monitor program used for monitoring time sharing terminal jobs. The terminal monitor program takes some action on the interruption caused by striking the SEND (or RETURN) key by a user. After the completion of the action it waits for the next interruption.

In this network system, AOMP and the user's job submitted by AOMP are permitted to run in a special job class. In this job-class, jobs start immediately after the submission. In addition, these jobs are given a higher priority in occupation of the CPU.

Operational example

A typical example which uses the back-end computer is the correction of closed orbit distortions (C.O.D. Corrections).[6] A series of NODAL programs manipulates the steps of the procedure. At first beam positions from 83 button monitors are obtained using the corresponding data module, and are displayed on a color graphic CRT mounted on the operator's console. These data (about 3.6 Kbytes) are sent to M-200H by the function FWARAY. The computer code PETROK, which is the KEK version of PETROS developed at DESY, is submitted by the function FSBMIT. PETROK calculates kick angles of the steering magnets and expected beam positions using the data of the beam position monitors and the basic parameters of the ring lattice already stored in the disk data-sets of M-200H and stores them on another disk data-set. The CPU time for PETROK is 10 seconds and the typical elapsed time is 30 seconds.

After the completion of the PETROK job, which is detected by the function FSTATU, the NODAL program reads back calculated kick angles and the expected beam positions using the function FRARAY. The size of these data amounts to 5.6 Kbytes. The operator judges if he actually loads the calculated steering excitation into the magnets or not by information appeared on the CRT display.

At last actual loading is carried out with the function of managing the power supplies of the steering magnets.

Acknowledgements

We wish to thank Professors Y. Kimura and G. Horikoshi for their encouragement and support during the work. We also thank the members of the TRISTAN control group and TRISTAN operation group for their stimulating discussion.

References

1) T. Nishikawa and G. Horikoshi: Status of KEK TRISTAN Project, IEEE Trans. Nucl. Sci. 30 (1983) 1983.
2) H. Koiso et al.: Computer Control System of TRISTAN, contributed paper to this conference.
3) Y. Asano et al.: KEKNET, A High Speed On-line Network for High Energy Physics, Nucl. Instrum. Methods 159 (1979) 7.
4) A. Akiyama et al.: KEK NODAL User's Guide, KEK Report 84-5 (1984).
5) M.C. Crowley-Milling and G.C. Shering: The NODAL System for the SPS, CERN 78-07 (1978).
6) H. Fukuma et al.: Correction of Closed Orbit Distortion in TRISTAN AR, contributed paper to this conference.