Development of a Control System for Stable Linac Operation

MASUDA Takemasa

DOCTOR OF PHILOSOPHY

Department of Accelerator Science
School of High Energy Accelerator Science
The Graduate University for Advanced Studies

2005

Abstract

The SPring-8 linac has to serve high stability of electron beams for a long period of a few months. It is because the linac is used for simultaneous top-up operation of the SPring-8 8-GeV storage ring and the 1.5-GeV NewSUBARU storage ring. Injections into NewSUBARU are very sensitive to variations of beam energy and beam trajectory. Effective tolerance of energy variations to keep high injection efficiency into NewSUBARU is presently 0.03% (rms). In order to achieve long-term stability of the linac electron beams, shot-by-shot data acquisition of non-destructive beam position monitors (BPMs) is required for investigation of beam instabilities and feedback control of the electron beams.

The old linac control system was built on the bases of standard technologies such as VMEbus, Ethernet, and UNIX computers. It had to be a base of the development of the BPM data acquisition system. Nevertheless, the old system didn't have enough control capability such as data logging system, high throughput of issued commands and so on. And the old system also couldn't provide enough stability to realize the long-term simultaneous top-up operations in both hardware and software.

As the groundwork for the stable linac operation and the shot-by-shot data acquisition, the new linac control system was designed and developed taking account of stability enhancement of the control system itself and availability enhancement of the linac operation. The system was developed through the extensive studies of modern operating system (OS), up-to-date electric circuit technologies using field-programmable gate array (FPGA), real-time control capabilities and so on. For the control software, MADOCA (Message And Database Oriented Control Architecture) was adopted. The MADOCA framework, which had been already used in the SPring-8 accelerator complex, provided satisfactory stability and powerful data logging system. Solaris was newly chosen as the real-time OS for Intel architecture VME CPU boards, according to the results of wide and profound studies of real-time functionality. The MADOCA framework was installed to the Solaris-operated modern VME CPU boards together with migrated driver software and device control application programs. The control hardware was radically renovated. Optical-linked remote I/O system (OPT-VME) and network-connectable pulse-motor control unit (MCU) were newly developed instead of direct I/O boards on the VMEbus. The new devices could keep the VME computers away from big noise sources, and could maintain the output signals even if the VME computers were rebooted. And the new devices could provide function-intensive control capability to the new control system.

As the result of the renovation, the new linac control system successfully brought high stability, high throughput and powerful data logging system. And the new control system never interrupted the SPring-8 simultaneous top-up operation in 2005 by its troubles, resulting in success of availability enhancement of the linac operation.

On the base of the new linac control system, the development of a shot-by-shot BPM data acquisition system came feasible. The new data acquisition system for the 47 BPMs in the linac was developed to

acquire all the 376 BPM data synchronizing to every beam shot. A set of synchronized BPM data was taken by six VME systems and a shared memory network. All the acquired dataset was stored into a general relational database (RDB). The data was recorded together with an event number and a time stamp for beam shot identification. The system successfully realized data acquisition with no detection losses at the 60pps linac operation using interrupt-based event detection, real-time feature of the Solaris, faster VME CPU boards and so on. The system was also designed to have real-time controllability to apply to fast feedback control in the future.

The archived BPM data in the RDB are utilized for automatic trajectories and energies corrections of the electron beams in every 5 minutes. The BPM data analysis successfully contributes to the achievement of long-term energy stability of 0.02% (rms) and beam positions stability of 30μm (rms) through the automatic feedback control.

The long-accumulated BPM data taken by the new system reproduces a past electron beam status. The analysis of the burst currents occurred in the linac was a good example to show the importance of the completeness of BPM dataset. An investigation of recorded BPM data brought a fruitful result to find a rare burst-current phenomenon. It was achieved by surveying a perfect dataset taken by the event-synchronized data acquisition system.

This study produces successful results in the development of the new linac control system and the new shot-by-shot BPM data acquisition system. The new control system greatly contributed to enhancement of the linac stability and the electron beam quality. In particular, the beam-synchronized data acquisition and the analysis of all the acquired data was effective approach for the linac stabilization. The new framework, furthermore, can be applied to 60Hz data acquisition of SCSS (SPring-8 Compact SASE Source), which is a coming 8-GeV X-ray free electron laser accelerator with C-band accelerating structures. The application will help the beam stabilization of the SCSS a lot.

Contents

1. Introduction **4**

1.1. Motivation of the development 4

1.2. Composition of the thesis 6

2. SPring-8 linac and accelerators control system **11**

2.1. SPring-8 linac 11

2.1.1. Overview 11

2.1.2. Energy compression system 12

2.1.3. Beam position monitor 13

2.2. SPring-8 accelerators control system 14

2.2.1. Hardware 14

2.2.1.1. VME computers 14

2.2.1.2. Field stations 16

2.2.1.3. Operator consoles 16

2.2.1.4. Server machines 16

2.2.1.5. Networks 17

2.2.2. MADOCA framework 18

2.2.2.1. Overview 18

2.2.2.2. Message Server 18

2.2.2.3. Access Server 19

2.2.2.4. Equipment Manager 19

2.2.2.5. Equipment Manager Agent 20

2.2.2.6. Poller/collector system 21

2.2.2.7. Database 24

2.2.2.8. Application programs 26

3. Development of the SPring-8 linac control system **41**

3.1. Background of the development 41

3.2. Methods of the development 42

3.3. Development of component technologies for the linac control system 43

3.3.1. Studies of the VMEbus CPU board 43

3.3.2. Studies of the operating system for the VMEbus CPU board 43
3.3.3. Development of the optical-linked remote I/O system 45
3.3.3.1. System design 45
3.3.3.2. Master and remote boards 46
3.3.4. Development of the network connectable pulse-motor controller 48
3.3.5. Development of the new connector-boxes 50
3.4. Software development of the linac control system 52
3.4.1. Software development for device control level 52
3.4.2. Software development for equipment control and man-machine interface levels 52
3.5. Installation of the new linac control system 53

4. Development of the event-synchronized data acquisition system for SPring-8 linac beam position monitors 83

4.1. Background of the development 83
4.2. Hardware of the data acquisition system 83
4.2.1. OPT-VME system 84
4.2.2. Shared memory network 84
4.2.3. VME computers 85
4.2.4. PC and database server 85
4.3. Software development for the event-synchronized data acquisition system 86
4.3.1. Development of the event-driven data acquisition software framework 86
4.3.2. Process synchronization with the shared memory network 87
4.3.3. Development of the data logging software 87
4.4. Performance measurements 88
4.4.1. Setup for the measurements 88
4.4.2. Measurements and results 89
4.4.3. Discussions 90
4.5. Studies of the performance improvements 90
4.5.1. Approaches to the performance improvements 90
4.5.2. Measurements and results 91
4.6. Applications for the beam stabilization in the -8 linac 92
4.6.1. Automatic beam position feedback control 93
4.6.2. Automatic beam energy feedback control 93
4.6.3. Investigation of unexpected burst current emissions 94

4.7. Future applications 95

5. Summaries and conclusions 117

Acknowledgements 119

References 120

Appendix 123

1. Introduction

1.1. Motivation of the development

In general, a linac is difficult to stably reproduce electron beams, especially beam energy. Difficulty of the energy stabilization depends heavily on acceleration mechanism of the linac. Because the electron beams are accelerated using radio frequency (RF) electric fields generated by a lot of high power RF equipment, the beam energy fluctuates with variations of RF power and RF phase. RF phase noise leads to short-term fluctuation, while room temperature drift causes long-term variation. Recently, synchrotron radiation (SR) experiments have been advancing to require more stable linac electron beams for a long period. Therefore, achievement of stable and precise device control system is essential much more than ever in order to obtain good reproducibility of electron beams.

In SPring-8 [1], which is the largest third-generation SR facility in the world, a 1-GeV linac works as the injector in the SPring-8 accelerator complex as shown in Fig. 1.1. The SPring-8 linac has to work for simultaneous “top-up operations” [2] of both the 8-GeV storage ring and the NewSUBARU storage ring [3]. The top-up operation is an ideal operation for the storage rings to keep the stored beam current approximately constant by injecting electron beams periodically. The stored beam current in the SPring-8 storage ring before and after the top-up operation is shown in Fig. 1.2. In order to achieve the simultaneous top-up operations, the stability of the electron beams during a few months operation are strongly required to the linac. In particular, the injection into the NewSUBARU storage ring is sensitive to the beam energy variation and beam positions drift. The permissible energy instability of the injected beam to NewSUBARU for stable injections is 0.03% (rms) [4]. As the results of many efforts, such as the stabilization of klystron drive line [5], the introduction of an energy compression system (ECS) [6], and the introduction of new synchronous timing system [7], the energy instability was successfully suppressed from 1% to 0.02% (rms). If the beam current of the linac was below 200mA, the instability is reduced to 0.01% (rms) [8]. Nevertheless, the long-term energy instabilities caused by the environmental temperature variations and unidentified drifts of beam trajectories were still left.

In order to investigate and suppress the instabilities, non-destructive beam position monitors (BPMs) were newly installed into the linac including the downstream beam transport lines until the summer of 2003. A fast real-time data acquisition system for the newly installed BPMs was required to the linac control system. The data acquisition system had to take all the BPM data synchronizing to the linac beam trigger at 60Hz that was the maximum repetition rate of the linac operation. All the acquired data, moreover, had to be stored into a general relational database (RDB) for off-line analyses of the beam instabilities and for feedback control for suppression of the beam instabilities. The BPM data acquisition system needed many advanced

component technologies like an event-driven software framework for synchronization between data acquisition software processes on the distributed computers, a fast network with real-time capability, a real-time OS for precise process control, fast data insertion method into a RDB and so on.

The old linac control system should be a base of the shot-by-shot BPM data acquisition system and the feedback control to the electron beams. However, the old system didn't have sufficient control capability to add the new function. In particular, lack of data logging system was the big problem. Also, the old system didn't have enough stability in both hardware and software to meet the long-term simultaneous top-up operations. Reboots or stops of VME computers sometimes occurred in the old system had interrupted the linac operation because some of output signals of VME I/O boards stopped due to the reboots. Many VME computers were placed near linac equipment along with the length of the linac. This location-oriented arrangement in the old system was not suitable for fast and tight control of common equipment deployed in the whole of the linac.

Therefore, drastic renovation of the linac control system was required as the groundwork for the stable linac operations and the shot-by-shot BPM data acquisition system.

In this study, the following methods for the development of the new linac control system were taken.

First, the control system itself should be stabilized. For the stabilization, the control system has to employ both well-designed software framework and control hardware with sufficient noise immunity. Second, the control system has to employ a data logging system to provide data analysis capability. Third, equipment control computers should have function-oriented arrangement to achieve fast and tight control of the common equipment distributed along the linac. Finally, the control system should be designed to provide high operation availability to the linac, that is, control system troubles shouldn't interrupt the continuous operations of the linac.

A framework MADOCA [9] was applied to the linac control software because MADOCA had satisfactory achievement of control in the SPring-8 storage ring, the booster and the NewSUBARU storage ring. For the control hardware, new component technologies such as an optical-linked remote I/O system and a network connectable motor control unit were developed to realize robust structure against the electric noise and to enhance operation availability of the linac. These new devices were useful to isolate the VME computers away from big noise sources. The devices were designed to keep a state of output signals, ex. origins of motor drivers, even though the VME computers were rebooted due to troubles. The device-state holding feature prevented interruptions of the continuous top-up operation from control system troubles.

The MADOCA framework was originally designed for the 8-GeV storage ring. The framework weren't adequate for real-time control that is necessary for shot-by-shot data acquisition of the linac BPM. Therefore, an event-synchronized fast data acquisition system was newly developed as function extension of MADOCA,

by considering its application for the SCSS (SPring-8 Compact SASE Source) linac, which is a coming X-ray free electron laser accelerator. The approach expanding the MADOCA framework brought easy debugging and high reliability to the data acquisition system. The BPM data archiving into a general RDB enabled reproduction of the past beam status and analyze the beam instability.

A shared memory network (SHM-net) was newly introduced as a data transfer backbone for the fast and real-time data acquisition at 60Hz, which is the maximum repetition rate of the SPring-8 linac and the SCSS linac. Ring-buffered data bank built on the SHM-net could hold the enough amounts of the acquired data for fast feedback control in the future.

The basic idea of this study has a general aspect. The idea was applied to the SPring-8 linac as the first case. Furthermore, it can be generalized to any other accelerator control system that has a different control framework.

1.2. Composition of the thesis

In chapter 1, motivations of this study and the composition of the thesis are described.

In chapter 2, overviews of the SPring-8 1-GeV linac and the SPring-8 standard control software framework MADOCA will be explained. Outlines of electron beam energy compression system (ECS) and non-destructive beam position monitors (BPMs) of the linac, also, will be described.

In chapter 3, development of the linac control system will be discussed. The development included hardware re-engineering and software upgrade in order to enhance the control system stability and availability. For the control hardware re-engineering, noise immunity (noise suppression) of the hardware system was enhanced for precise monitoring of equipment signals and stabilization of the hardware system itself. And for the availability enhancement of the linac operation, the hardware system was updated to keep output signals even if the VME computers were rebooted. Component technologies necessary for the hardware re-engineering will be mentioned from the viewpoints of noise immunity enhancement of the VME systems and availability enhancement of the linac operation. For the software upgrade, the MADOCA framework was adopted to the new linac control system by introducing Solaris-operated Intel architecture VME CPU boards. The studies of operating system (OS) and VME CPU boards will be discussed from the real-time point of view [10]. The development of the new linac control system will be reported together with a summary of operation results.

In chapter 4, development of an event-synchronized data acquisition system for the new BPMs will be described. As already mentioned in the above, the BPM systems were newly installed in the linac in order to investigate the beam dynamics features, and to apply the acquired beam position data to the feedback control of the electron beams [11]. Methods of the development and studies of the performance improvement will be discussed. The details of data acquisition system succeeded in shot-by-shot data acquisition at 60Hz

operation will be given. The description includes the accumulated data structure, a shared memory data transfer network and a distributed relational database.

There are similar applications of BPM data collection systems using a shared memory network in other laboratories such as Advanced Photon Source (APS) [12] and Taiwan Light Source (TLS) [13]. In their applications, all the systems are designed for global fast feedback for storage ring orbit corrections, and the acquired data are not accumulated in a database. The greatest advantage of the unique data acquisition system developed in this study is that all of the beam position data of all beam shots up to 60Hz is accumulated into the database without any loss. Analysis of the unexpected burst current that happened very occasionally in the SPring-8 linac will be described as one of the application examples of the advantage. Of course, the new framework for the shot-by-shot data acquisition was designed to perform global fast feedback in the linac.

The measurement of the 60Hz synchronizing data acquisition and the data-taking system will be described, focusing on the new idea and the software architecture. The results of BPM position measurements by use of 1-GeV electron beams of linac will be discussed, and successful feedback control to suppress the electron beam instabilities by using the acquired beam position data will be mentioned as well.

An application of the event-synchronized data acquisition system to the SCSS 8-GeV linac will be described as the future application.

In chapter 5, the results obtained in this study will be summarized together with conclusions.

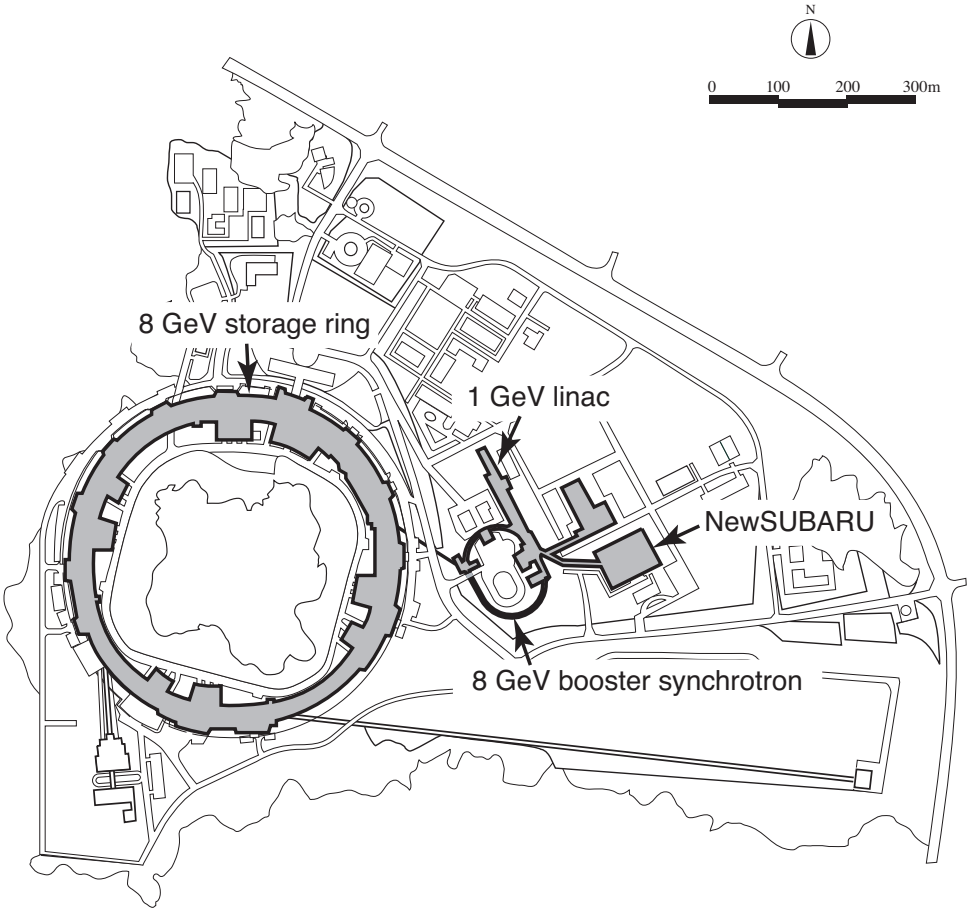
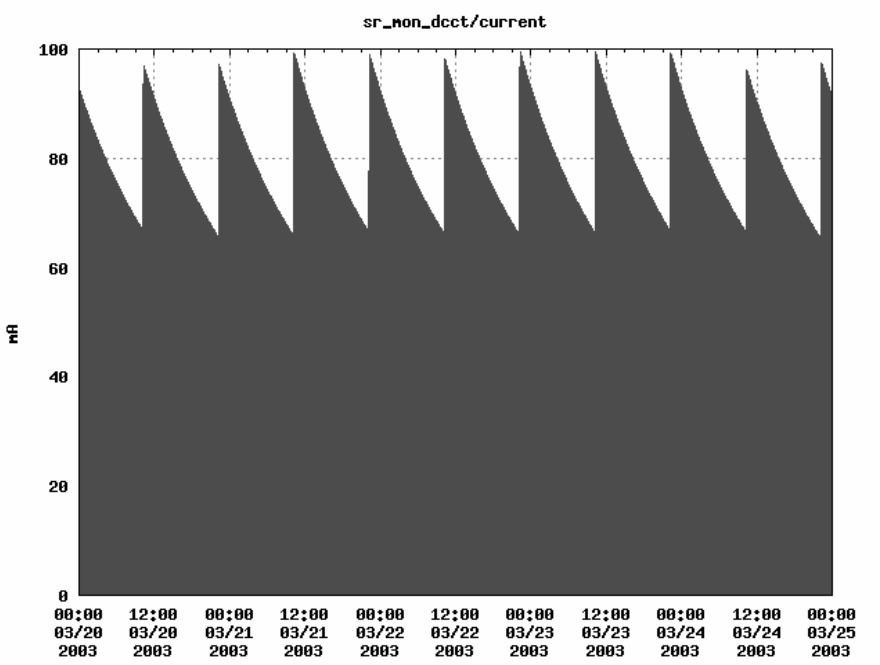
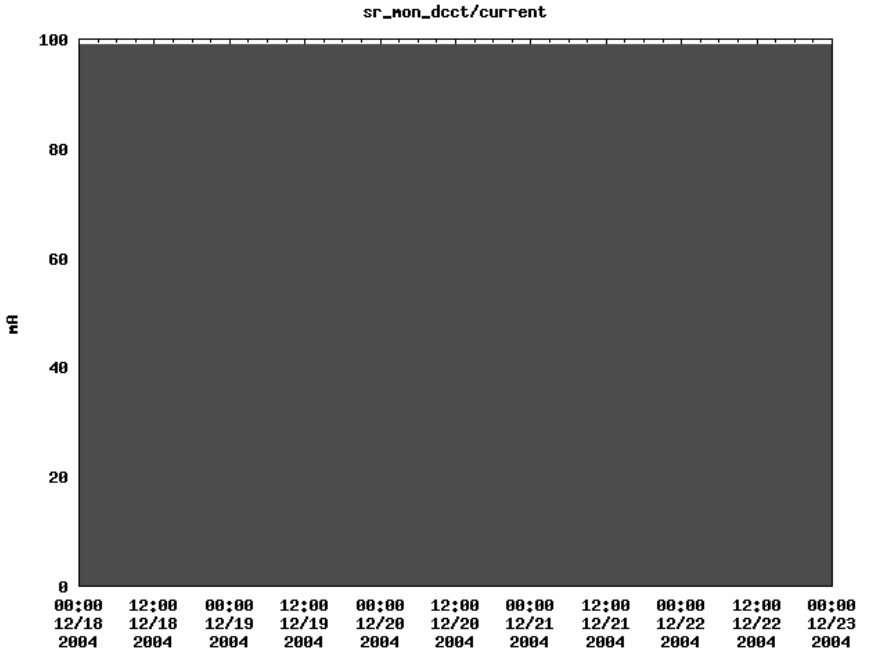


Fig. 1.1. An overview of the SPring-8 accelerator complex.

(a)


sr_mon_dcct/current
100
80
60
40
20
0
mA
00:00 03/20 2003
12:00 03/20 2003
00:00 03/21 2003
12:00 03/21 2003
00:00 03/22 2003
12:00 03/22 2003
00:00 03/23 2003
12:00 03/23 2003
00:00 03/24 2003
12:00 03/24 2003
00:00 03/25 2003

(b)


sr_mon_dcct/current
100
80
60
40
20
0
mA
00:00 12/18 2004
12:00 12/18 2004
00:00 12/19 2004
12:00 12/19 2004
00:00 12/20 2004
12:00 12/20 2004
00:00 12/21 2004
12:00 12/21 2004
00:00 12/22 2004
12:00 12/22 2004
00:00 12/23 2004

(c)

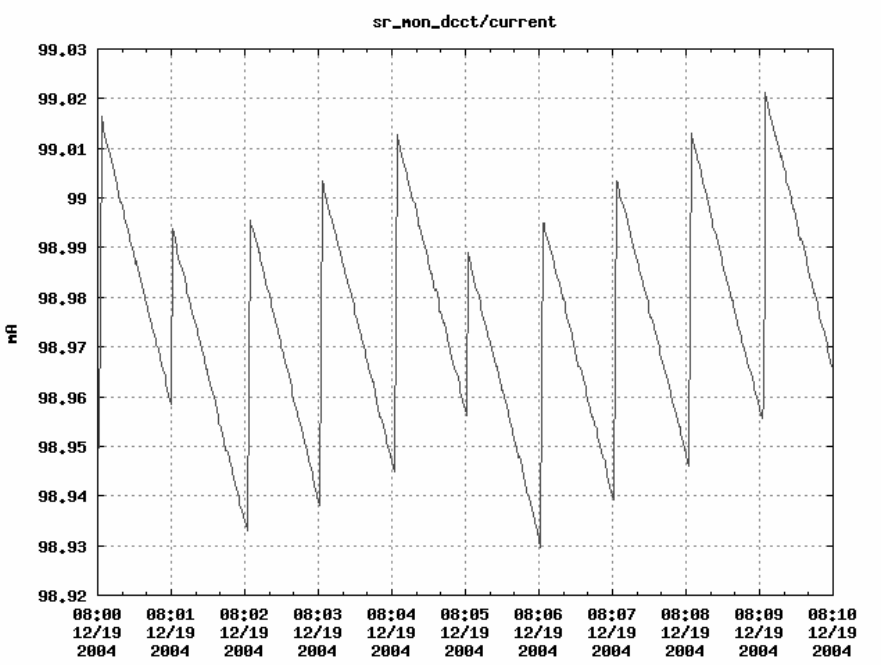


Fig. 1.2. The stored beam current in the SPring-8 storage ring before and after the top-up operation. Fig. (a) shows the stored beam current variation before introducing of the top-up operation. At that time, electron beams were injected twice in a day. Fig. (b) shows the stored beam current with the top-up operation, and Fig. (c) is a magnification of a part of the Fig. (b). It indicates electron beams were injected at one-minute interval.

2. SPring-8 linac and accelerators control system

2.1. SPring-8 linac

2.1.1. Overview

The SPring-8 linac is a length of 140m, and accelerates electron beams up to 1 GeV. The linac consists of a thermionic electron gun, a bunching section, an accelerating structure for a 60-MeV pre-injector, a main accelerating section and an energy compression system (ECS). The linac components are installed on the first floor of the linac housing. The accelerated beams are transported for further acceleration and utilization via three transport lines. The transport lines, linac-to-synchrotron beam-transport line (LSBT), linac to accelerator R&D facility (L3BT), and linac-to-NewSUBARU (L4BT) are connected to downstream of the main acceleration section. Fig. 2.1 shows an overview of the linac and the three beam-transport lines. The present beam parameters for injections to the booster synchrotron and the NewSUBARU are summarized in Table 2.1.

The thermionic electron gun, which operates with 190kV between a cathode and an anode, can provide maximum peak current of 5.7A. The emission current can be controlled by adjusting grid bias voltage and inserting a beam iris located at downstream of the anode. Two different types of grid pulsers are prepared in order to generate beam with different pulse length of 1nsec and 40nsec, which are required depending on the filling pattern of the storage ring.

The electron beams from the thermionic electron gun are bunched into 10ps (FWHM) or less, and are accelerated up to 9MeV by two pre-bunchers and a buncher in the bunching section. The pre-buncher is a re-entrant type single cell cavity operated at 2856MHz. The pre-bunchers bunch the electron beams into 50ps (FWHM) length. The buncher is a 13-cell side-coupled cavity operated at 2856MHz.

The bunched beams are accelerated up to 60MeV by a 3-m long accelerating structure in the 60-MeV pre-injector section (H0). This accelerating structure is same as that of a main accelerating section.

The main accelerating section has 24 sets of accelerating structures, and accelerates the electron beams up to 1 GeV. The accelerating structure is a constant gradient traveling wave-type, operating in the $2/3\pi$ mode at the operation frequency of 2856MHz. A maximum RF power supplied to an accelerating structure is 35MW, and an accelerating field gradient reaches 18MV/m.

The block diagram of the present linac RF system is shown in Fig. 2.2. Thirteen sets of 80MW klystrons are installed in a klystron gallery in order to supply RF-power to the pre-bunchers, the buncher, and 26 accelerating structures. The klystron gallery is on the second floor of the linac housing. Klystron modulators are installed along the klystrons on the same floor. The most upstream klystron supplies the RF power not

only to the two pre-bunchers, the buncher and one accelerating structure in the H0, but also to an RF drive-line used for the 12 sets klystrons in the main accelerating section. Each klystron receives the RF power from the drive-line through an attenuator and a phase shifter, and supplies the output RF power to the two accelerating structures. The attenuators are used to optimize the input power for the klystrons, and the phase shifters are used to adjust the phase difference between electron beams and RF field. The most downstream klystron supplies RF power to the accelerator structure in the ECS in addition to the two accelerating structures in the main accelerating section.

Quadrupole (steering) magnets installed in each section are used to adjust beam size (position), respectively. Bending magnets are used to switch beam route for the specified transport line. In order to realize simultaneous top-up operations both for the 8-GeV storage ring and the NewSUBARU, an LSBT bending magnet was replaced to a lamination type magnet. By using the lamination magnet and a newly installed fast-response power supply, rise time to the maximum magnetic field strength of 0.9T is shortened from a few seconds to 0.2sec [15][16].

Fluorescent-screen beam-profile monitors are installed in each accelerating section and beam transport lines to measure beam position and size. Strip-line type beam position monitors (BPMs) are also installed in each accelerating structure and beam transport lines to measure beam position without giving interference to the beams. These monitors are indispensable to carry on beam dynamics study for the stabilization of beam energy and injection efficiency.

2.1.2 Energy compression system

The ECS is installed in the downstream of the linac in order to suppress energy spread of the out going beams to the booster and the NewSUBARU [7]. The ECS consists of a 9.2-m long chicane section and a 3-m long accelerating structure which is the same type used in the main accelerating section.

The chicane section comprises two correction quadrupole magnets, and four bending magnets which form a chicane orbit. Bending magnet is a rectangular type magnet which has a length of 1.7m, a bending angle 24°, and a bending radius 4.1m. Two quadrupole magnets are placed between the second and third bending magnets in order to satisfy dispersion-free condition in a matching section at the downstream of the ECS. The maximum energy dispersion in the chicane is $\eta = -1$m.

Fig. 2.3 shows schematic view of the ECS components and energy compression process at the ECS. When a bunched beam with an energy spread passes through the chicane, electrons in the bunch travel in different orbits depending on their energy. Electrons with higher (lower) energy travel shorter (longer) orbits, and shift the positions to the head (tail) in the bunch, respectively. Consequently, the bunch length is longitudinally extended depending on the energy deviation after passing the chicane. Then, RF power which has a phase to decelerate the head-part electrons and accelerate the tail-part electrons was given to the shaped bunch in the accelerating structure at the downstream of the ECS to suppress the energy spread. Since the ECS

compresses the energy spread into the beam energy at the zero-crossing point of the RF field in the accelerating structure, the ECS is available to adjust the central energy of the beam precisely by changing the RF phase.

It is essential for the ECS to stabilize the RF phase of the accelerating structure because the RF phase shift sensitively causes the central energy drift of the electron beam. In order to stabilize the RF phase, a phase lock loop (PLL) is installed in an RF drive-line for the most downstream klystron. The PLL eliminates temperature dependences of a 120m long coaxial cable for the ECS drive-line.

The ECS has an optical transmission radiation (OTR) monitor between the second and third bending magnets in the chicane section, in order to measure the central energy and energy spread of beams before the ECS compression. The OTR screen is made of a 12.5mm thick kapton foil with 0.4 mm aluminum coating. Since the 1-GeV electron beam passes through the OTR screen without any loss, the OTR monitor is always available for the beam profile measurement during injections to the booster and the NewSUBARU.

2.1.3 Beam position monitor

In order to measure beam positions non-destructively and to enable correcting beam trajectory and energy, 47 BPM sets had been installed in the linac and three beam-transport lines by the summer of 2003 [17][18]. From linac operation requirements, the BPM system was designed to achieve position resolution to less than 0.1mm at full width (±3σ). The signal detection was also required to have a wide dynamic-range since the beam current varies widely depending on the beam pulse width. Fig. 2.4 shows a schematic diagram of the linac BPM system. One BPM system consists of a four-channel electrostatic strip-line monitor, a four-channel band pass filter (BPF) module, and a four-channel detector module [19].

The strip-line monitor has 27mm long strips and two kinds of aperture. A strip-line monitor installed in a non-dispersive section has a circular aperture of 32mm diameter. On the other hand, a monitor in a dispersive section has an ellipse aperture of 62×30mm. Beam positions were measured by the monitors with the detection frequency of 2856MHz.

The signal processor consists of two NIM (nuclear instrumentation module) modules, i.e. the BPF module and the detector module. The center frequency of the BPF module is 2856MHz, with a bandwidth of 10MHz. In each channel of the detector module, a demodulating logarithmic amplifier AD8313 [20] is used to detect the S-band RF signal, consistent with the requirement for a wide dynamic range in the beam currents to be measured. The signal from the logarithmic amplifier is further processed with a self-triggered peak-hold circuit and an externally triggered sample-hold circuit, and then converted to a digital output with 16 bits of resolution.

Using an output voltage of each electrode, position data at a BPM are calculated as follows:

$$X = C_x(\log_e A - \log_e B - \log_e C + \log_e D), \qquad \text{..... (2.1)}$$

$$Y = C_y(\log_e A + \log_e B - \log_e C - \log_e D), \qquad \text{..... (2.2)}$$

where

X: horizontal beam position,

Y: vertical beam position,

C_x, C_y: proportional coefficient,

A, B, C, D: output voltage of each electrode.

2.2. SPring-8 accelerators control system

Accelerator control systems of the linac, the booster, and the storage ring were separately developed due to the different construction schedule. The SPring-8 standard control system, MADOCA, was originally designed and developed for the 8-GeV storage-ring control.

Common design concepts of all the sub-system were as follows:

(1) Adopt so-called “standard model (3-tier control structure)” as the system structure.

(2) Operate all the accelerators at central control room by small number of operators.

(3) Build the system by using industry standard hardware and software as much as possible.

A structure of the “standard model” consists of three layers of a presentation layer, an equipment-control layer, and a device-interface layer.

In the presentation layer, UNIX workstations were adopted for man-machine interface of GUI programs based on X11. They were installed in the central control room, and connected with high-speed backbone network and were communicated with remotely distributed VME computers. The VME computers controlled accelerator equipment as front-end controllers. Server machines used as database servers and file servers were also connected with the backbone network in the central control room [21].

The framework MADOCA was designed with a client/server scheme because the client/server scheme was promising to provide redundant software architecture, rapid development of application programs, and higher degree-of-freedom of the software design [9][22]. Accelerator operation programs were built on the basis of message-oriented middleware of the MADOCA. The accelerators were controlled by issuing man-readable control messages from the client applications to the server processes running on the VME computers. The middleware manages handling of the control messages and network communication between the client processes and the server processes. The message-oriented middleware of MADOCA helps the application programmers to program easily and accelerator operators to understand the equipment operations.

2.2.1 Hardware

2.2.1.1 VME computers

VME computers have been adopted as front-end controllers in all the accelerator control systems. Since the VMEbus system provides high reliability, expandability and flexibility, it has been widely applied to a front-

end controller as an industrial standard. It has up to 20 slots in one chassis, and many commercial CPU boards and I/O boards are available. The VMEbus system provides capability of multi-masters (multi-CPU boards) configuration.

In the storage-ring control system, an HP9000/743rt [23] CPU board was employed as a VMEbus controller at first [24]. It was powered by PA-RISC 7100LC [23] operated with 64MHz clock, and the performance of the CPU was 77.7MIPS. It had a 16MB main memory, and a 20MB PCMCIA flash disk card which was used as a boot device. The CPU was operated by HP-RT [23] OS. HP-RT was PA-RISC version of LynxOS [25] which was real-time UNIX made up from scratch. Since the 743rt CPU board had been discontinued in November 1999, and there were not any other platforms available to run HP-RT, the 743rt and HP-RT system was replaced by a system consisting of Intel-Architecture (IA-32) based CPU board and Solaris OS [26].

All the VME computers boot the OS from a local flash disk, and mount a common storage via Network File System (NFS). An NFS server machine exported disk files for application programs.

All clocks of the VME systems were synchronized to a *master* clock with Network Time Protocol (NTP) software. The master clock was a NTP server machine in a machine LAN. Because HP-RT only did not support the NTP software, an original method was developed to adjust a clock of the HP-RT system. This scheme adopted the client/server architecture, and all the clocks of the HP-RT systems were adjusted to the clock of an operator console synchronized to the first stratum NTP server.

Direct I/O boards, i.e. digital-input (DI) boards, digital-output (DO) boards, TTL-level digital-input/output (TTL DI/O) boards, analog-input (AI) boards, and pulse-train generator (PTG) boards were used in the storage ring VMEbus systems. Specifications of the boards are listed in Table 2.2.

Two field-buses which were attached to the VMEbus were also adopted in the storage-ring control system. One was GP-IB bus, and the other was a remote I/O (RIO) system. The RIO system [27] was initially developed for magnet power-supply control [28], and eventually applied to the storage ring BPM data acquisition and vacuum-equipment controls. The RIO system provides good electric isolation and is robust against the noise. It consists of RIO master boards, six kinds of remote I/O boards (type-A, B, C, E, F, G), and one-to-eight optical-linked multiplexer boards. The features of the RIO system are as follows:

- Optical fiber connection up to 1km transmission distance.
- Serial link communication between master and slave boards.
- One master board can control maximum 62 slave-boards.
- 1 Mbps transmission rate.
- HDLC protocol for communication between master and slave boards.
- A twisted pair cable with RS-485 interface is available for slave connection.

Fig. 2.5 shows an overview of the RIO system, and Table 2.3 shows the specifications of all the RIO slave boards. The number of VME computers for a large control system can be reduced by introducing the RIO system.

2.2.1.2 Field stations

As a supplementary system of the VME computers, a Linux-based PC system was deployed for temporary measurement. In general, PCs are not reliable enough when comparing with VME systems, but PCs are more inexpensive and powerful than VME computers. And many kinds of inexpensive commercial I/O boards with Linux device drivers are available. PCs contribute to early start-up of measurement systems.

Since the SPring-8 standard software framework was already migrated to Linux, the Linux base PC, called Field Station, provided almost the same functions as that of the VME system except for deterministic process control [29]. Many I/O devices from ISAbus and PCIbus, for example, digital I/O boards, analog I/O boards, GPIB-ENET109 [30] controllers which work as Ethernet-to-GPIB converters, were available.

2.2.1.3 Operator consoles

As operator consoles, 17 workstations currently work in the central control room. They are eleven B2600 workstations (500MHz PA-8500), one J6000 (2 × 552MHz PA-8600), and six J6700 workstations (2 × 750MHz PA-8700). An operating system, HP-UX [23] 11.0, runs on all the workstations. Application programs such as accelerators operations, automatic orbit corrections, equipment controls, periodic data acquisition, alarm surveillance, and alarm display are running on the consoles.

All the consoles mount common NFS file systems exported by the NFS server machine and a database-server machine. An NTP scheme made all the system clocks of consoles synchronized to the first stratum NTP server in a machine LAN. All the time stamp of the consoles synchronized within 10msec.

2.2.1.4 Server machines

A relational database management system (RDBMS), Sybase Adaptive Server Enterprise (ASE) [31], is used for the SPring-8 database servers. Database server machines need much computer resources such as CPU power, memory, network bandwidth, disk size and disk-access bandwidth. In particular, high reliability was strongly required. Since the SPring-8 had started the operation in 1997, the database server machine was upgraded in several times to process a number of increasing signals. As the latest server system, a high availability (HA) cluster has been built by using two server machines. One is an HP9000/rp4440 server that has eight way PA-8800 CPUs with 800MHz clock, 8GB memory, and two internal 73GB SCSI disks mirrored by mirror-UX software. The other is an HP9000/N4000 server that has five way 550MHz PA-8600 CPUs, 4GB memory, and two internal 36GB SCSI disks mirrored by mirror-UX [23] software. HP-UX 11i operating system is used for the database servers. The servers share two mirrored disk enclosures of an

amount of 648GB volume that are linked by Ultra2 Low Voltage Differential (LVD) SCSI interfaces. The disks are used as raw devices to achieve fast access by database server processes. Network interfaces and power supplies of both server machines are also duplicated for the system redundancy. Cluster management software MC/Service Guard [23] watches status of the cluster. When the MC/Service Guard detects fault of hardware parts, it switches the parts to the backup ones.

Usually, the HP9000/rp4440 (HP9000/N4000) works as a main (stand-by) server, respectively. If the main server is down, then the stand-by server takes over the services in 1 or 2 minutes, i.e. failover. Application programs such as machine operation GUIs automatically re-connect to the stand-by server, with help of Sybase ct-library to access the database.

A dedicated cluster server is employed to keep an archive database which stores log data since the SPring-8 commissioning. The cluster server for the archive database works as a remote database server of the main database. The cluster server consists of two DELL Power Edge 6650 server machines that are equipped 4 × 2GHz Intel Xeon CPUs. The archive servers run on the operating system Red Hat Enterprise Linux Advanced Server (AS) 2.1. The Red Hat cluster provides middle level of HA operation. Once the server machine on which database server is running fails, the stand-by server machine automatically takes over the database. However, application software, which is connecting to the database, has to re-connect to the database by manually, because the connection to the database is lost by the failover.

Two HP9000/A500 server machines form a high availability NFS server in the control LAN. They have a 550MHz PA-8600 CPU, 512MB memory, and two internal 18GB SCSI disks that are mirrored by software. The server machines share two external disk-enclosures connected with Ultra2 LVD SCSI interfaces. When the SPring-8 operation started in 1997, one operator console played a role of a file server for other operator consoles and VME computers. However, in order to avoid heavy load to both application programs and NFS server process, a dedicated NFS server machine has been introduced.

2.2.1.5 Networks

A duplicated 100Mbps FDDI network with dual-ring topology was adopted as a backbone network [33], as shown in Fig. 2.6. All the accelerators had own FDDI backbones, and all backbones were inter-connected to a FDDI switch in the central control room. Consequently, each FDDI backbone provided maximum bandwidth of the FDDI (100Mbps). All the VME computers were connected to the FDDI backbones through layer-3 switching HUBs of FDDI nodes. The storage-ring backbone equipped four layer-3 switching HUBs, so that total seven layer-3 switching HUBs were used [34]. Optical fibers were used for connection between the FDDI nodes and the VME systems in order to avoid influence of the electromagnetic noise.

For tight security of the machine LAN, network firewall machines were introduced between the machine LAN and a laboratory public LAN. Normally, nobody can access the machine LAN from the public LAN via the firewall.

2.2.2. MADOCA framework

2.2.2.1 Overview

Fig. 2.7 shows a schematic diagram of the SPring-8 standard control framework MADOCA [9]. The MADOCA framework is based on the client/server architecture, and it provides message-oriented middleware to control equipment with a command message of a 255-characters string.

The MADOCA framework consists of a group of software Message Servers (MS), Access Servers (AS), Equipment Managers (EM), poller/collector cyclic data acquisition system, and databases. Basically, a set of a MS and several ASs works as the middleware for a local communication on the operator consoles. The local communication uses message scheme provided by System V UNIX. The AS makes multiple communications to the EMs running on the remote VME computers using ONC/RPC (Remote Procedure Call).

The control message has an English like S(subject)/V(verb)/O(object)/C(complement) format. The character string is called as an S/V/O/C message. A client process, represented by the S, sends a control message to a server process that manages equipment object represented by the O. The term S is identified by a combination of a process ID, a process name, a user name, and a hostname. The term V means a control action to the O. Ordinarily, "put" and "get" command are used as the V. The "put" command is used to set value represented by C. On the other hand, the "get" command is to acquire data represented by the C. The term O is not a device channel nor slot, but means an abstracted equipment object (ex. power supply) which is controlled by an operation command. The term C represents a property of the equipment object O.

For example, a control message "*123_maggui_operator1_console1/put/ring_magnet_powersupply_1/12.3A*" means that a process named *maggui* with process ID *123* and account *operator1*, running on the hostname *console1*, requests a magnet power supply named *ring_mag_powersuppliy_1* to set a current to *12.3A*. And a control message of "*123_maggui_operator1_console1/get/ring_magnet_poweersupply_1/current_adc*" means that the same process request to the same magnet power supply to get an output current to be obtained by an A/D converter.

The MADOCA framework has adopted a device abstraction concept. The control message is abstracted in the accelerator equipment level, so that application programmers for machine operations need not to know the details of the VME computers, i.e. I/O boards, channel numbers, and other physical configurations.

2.2.2.2 Message Server

An MS process runs on each operator console, and it plays a role of message distributor to local processes. Ordinarily, the MS receives a control message sent by an application program such as operation GUIs. After checking message format, destination, and privilege, it forwards the message to the destination such as an AS.

Then MS waits to receive a reply from equipment through the AS. And then it forwards the reply to the source application process which sent the message.

The message scheme has a queue, and the queue is a FIFO (First-In First-Out) structure. The UNIX system defines the message queue size, which is possible to change. Currently, the queue size is defined to the maximum value 64KB, so that the maximum 229 messages can be stored in the queue. Each application is assigned an ID number called mtype in order to identify the messages. Since a message is tagged by an mtype, the MS can send the message to the proper destination application.

When the MS starts, it reads an access control list (ACL) file as shown in Fig. 2.8. The ACL file defines relations between object (O) names and destination process names, and has lists of accounts that have privileges to handle the message.

The message transaction speed of the MS running on the operator console is found to be less than 1msec. This speed is smaller than a speed of a network communication between the AS and the EM.

2.2.2.3 Access Server

The AS is a server process to manage network access between the operator consoles and the VME computers. It runs on the operator consoles and communicates with EM processes running on the remote VME computers using RPC. One AS process is prepared for each equipment group such as magnet power supplies, RF system, vacuum system, beam monitor system and so on. Only one AS of each equipment group runs on each operator console. After the AS receives a control message from the MS, the AS forwards the message to the related EM. Then the AS waits and receives the control result from the EM, and forwards the reply message to the MS.

When the AS starts, it retrieves equipment information of the related group from a database. The information includes relations between object (O) names and hostnames of VME systems to which the control messages should be sent. When the AS receives a control message, it parses the message and determines the destination VME from the equipment information.

A typical speed of round trip communication between the AS and the EM is about 10msec, where execution time of the EM is not included. This result was measured with an HP9000/743rt and 10Mbps Ethernet configuration.

2.2.2.4 Equipment Manager

EM is an RPC server process to realize device abstraction concept [35]. The EM process runs on each VME computer and waits S/V/O/C control messages sent by the AS processes via a network. The EM interprets a control message abstracted in equipment level, and controls I/O boards on the VMEbus.

First, the EM parses a received S/V/O/C message and translates the abstracted command to control of VME I/O boards, which is called an interpretation process of the EM. Next, the EM executes actual controls

of boards by specifying I/O channel, and gets a binary data from the board, which is called a control process of the EM. Finally, the EM translates the result of the I/O board control into an abstracted result in the equipment level, and returns the result to the application program which sent the control message. When the EM receives a control message, it always executes these three processes, i.e. an interpretation process, a control process and an abstraction process.

All the relations between S/V/O/C commands and control instructions, I/O channel definition, and calibration constants are described in a device configuration table called *config.tbl*. When an EM program is initialized at first, the EM reads a *config.tbl* and keeps it on an internal memory. A basic format of the *config.tbl* is explained in Fig. 2.9. In the 1st layer, the S/V/O/C commands are classified by V/O elements. In the 2nd layer, the S/V/O/C commands with the same V/O element are classified by the C elements. And in the 3rd layer, function names and arguments for interpretation, control and abstraction procedures are described. The function name of the control procedure generally has arguments of device files to access the I/O boards (ex. /dev/di_0). Fig. 2.10 shows an example of a *config.tbl* file. In this example, when the EM receives a message "*S/set/sr_mag_ps_st_v_1_1/123.5A*", the EM calls *em_mag_st_conv_put()* function with arguments of *1*, *6.5535e+3* and *3.27675e+4* for an interpretation procedure. Here S is an application process name which sends the "*set/sr_mag_ps_st_v_1_1/123.5A*" message. Then, the EM calls *em_mag_st_current_put*() function with arguments of */dev/rio_0* and *1* for a control procedure. And then the EM calls *em_std_ret()* function for the abstraction procedure. If the control succeeds, the EM returns a message "*sr_mag_ps_st_v_1_1/set/S/ok*" to AS, and if the control is failed, a message "*sr_mag_ps_st_v_1_1/set/S/fail*" is returned.

An EM holds a connection counter with AS processes. When the EM receives a connection request from an AS for the first time, the EM is initialized and calls a special function for initialization, then increases an internal counter from zero to one. In this special function, the EM may initialize the I/O boards and pre-process the equipment before EM receives the first control message. When the EM is disconnected by the AS, the EM is terminated and calls a special function for the termination, then the internal counter changes from one to zero.

2.2.2.5 Equipment Manager Agent

For the purpose of feedback control by software, a stand-alone process named equipment manager agent (EMA) is prepared in the MADOCA framework [36]. An EMA is created as a daemon process by an EM and communicates with the EM through an MS. The EM controls the EMA by using S/V/O/C control messages, which have the same format as that of the real equipment control. The EMA can be interpreted as a pseudo device made by software.

The EMA consists of two parts. One is a common frame that manages a control of the EMA process such as start/stop and feedback parameter setting for the EMA. The other is a feedback algorithm which

manipulates VME I/O boards by using the existing EM framework. Since the EMA is developed with the same framework as that of the EM, the same functions and *config.tbl* for the EM are available for the EMA. For example, the EM sends S/V/O/C commands by recursive call of API functions prepared for RPC clients, also the EMA can recursively send the S/V/O/C commands in the same way. The function of recursive calls of S/V/O/C commands contributes a lot to save EMA development time.

When the EM receives a *create* command for an EMA from a GUI program, the EM creates the specified EMA process which uses the same *config.tbl* file as that of the EM. Then, the EMA process reads the *config.tbl,* and opens a connection to MS which is already running on the same VME computer. The EMA and the EM can communicate with each other through the MS. When the EMA receives *start* command, it repeats the specified control sequence until it receives the *stop* command. At the beginning of each execution of the control sequence, the EMA checks a message from the MS. If EMA receives a message (command) from the EM, it executes the received command and replies a result to the EM. A feedback sequence is made of a combination of S/V/O/C commands, which are recursively executed with the EM functions, and is coded in a user-defined function. The EMA process stops by receiving a *destroy* command.

Since the EMA performs a feedback-loop on a VME computer, it can provide the faster control than a GUI control through network communication. As an example of the EMA applications, the EMA scheme is applied for a klystron control. In order to ramp up the klystron power, klystron EMA repeats a feedback sequence that reads vacuum gauges and adjusts the klystron power. During the ramping up of the klystron power, an arc may occur in the cavity due to non-flatness of the cavity inner surface. If the arc hits the surface and kicks out gases, the cavity vacuum pressure would become worse and the reflection power to the klystron would increase. In this case, the EMA tunes the klystron power. After the vacuum pressure would be better, the ramping up of the klystron is restarted. In the beginning, this feedback loop was performed in the GUI program level by sending the S/V/O/C commands to the EM over the network. It took the time of 300msec for one loop of the sequence including return message handling in the GUI programs. It was too slow to detect the vacuum pressure becoming worse. By introducing the EMA scheme, the sequence time was reduced to about 1/10, providing stable operations and smooth ramping up of the power.

2.2.2.6 Poller/collector system

Periodic data acquisition software, called a poller/collector system, is prepared to monitor accelerators status efficiently [37]. The system periodically collects equipment data and stores it in an on-line database. All the collected data can be monitored by retrieving from the on-line database. The poller/collector data-acquisition system consists of three parts, i.e. poller (Poller) processes, collector server (CS) processes, and collector client (CC) processes. The Poller processes running on the VME computers read equipment data sequentially and store it in a shared memory. A shared memory is one of the internal process communication way provided by SystemV UNIX (IPC-SMH for short). The CS on VME takes the data stored in the IPC-

SHM and sends it to the CC by a request. The CC process running on the operator console collects all the data from the CS, and inserts the data into the on-line database.

Poller

The role of the Poller process is to acquire equipment data cyclically. One or more Poller processes run on the VME computers. The Pollers are created by the CS by being given a data taking start message from the CC. The number of Poller processes is defined according to the number of data acquisition cycles. For example, if there are signals updated with 1sec and 5sec intervals in one VME computer, two Pollers have to be prepared on the VME.

After the Poller starts, it reads a Poller/Collector management file (PCMF) prepared for each VME computer. The PCMF is created from the parameter database, and the PCMF has the corresponding VME hostname as the filename. The PCMF contains information related to the CS and the Pollers, such as the executable filenames of the Pollers, polling cycles, and a list of signals, as shown in Fig.2.11. The file has tag format like XML (eXtensible Mark-up Language). One set of <collectorserver> and </collectorserver> tag provides properties for the CS. <poller></poller> tags define properties for the Pollers, and <signal></signal> tags define the acquired signals as the same manner. In each tag, properties of the CS and the Pollers can be defined by giving some elements. The PCMFs is placed in particular directories on NFS exported by the file server, and all the VME computers and operator consoles have to mount the directories.

The Poller employed the EM framework. While the AS calls the EM APIs as RPC via a network, the Poller calls the EM APIs as local function calls. All the S/V/O/C commands to be executed by the Poller are listed in the <signal></signal> tags in the PCMF. The Poller sequentially executes the S/V/O/C commands by calling the EM APIs to acquire the signals. This means that it is not necessary to develop new functions for the Poller once the EM running on the same VME computer has been developed. It saves the software development time. The same *config.tbl* file and same user-functions as the EM are available for the Poller. The user-functions for the EM are statically linked to the Poller at the build time.

The Poller process stores execution results of the S/V/O/C commands in an IPC-SHM which has a ring-buffer structure. One IPC-SHM is prepared for each Poller in a VME computer. A set of acquired signals forms one record, and a record size of the ring-buffer is given in a parameter database. The parameter of the record size is reflected as *ringsize* property in each <poller></poller> tags in the PCMF. Fig. 2.12 shows a structure of the IPC-SHM. As soon as the Poller updates the record in the ring-buffer, it also updates the newest record number and the newest acquired time in a header area of the IPC-SHM. The CS process monitors the newest acquired time in the header to check whether the Poller is working or not.

If the execution result of the S/V/O/C command is failure, the Poller sets a fail data flag in the ring-buffer instead of the “fail” string. If a signal is defined as “not in active”, the Poller sets an off data flag in the ring-buffer. A definition to collect a signal or not is given in the parameter database, and the collection status is

specified in the PCMF as an *action* property of the <signal></signal> tags for the signal. Table 2.4 represents the definitions of the fail data flag and the off data flag for each data type.

Collector Server

A CS runs on the VME computers and works as a server process of a CC process running on an operator console. Main tasks of a CS are data collection from IPC-SHMs and management of Pollers.

When a CS process receives initialization request from a CC, it reads a PCMF and then creates Poller processes and IPC-SHM for each Poller according to the PCMF file. When the CS cyclically receives data-collection request from the CC, it collects the newest set of data in the IPC-SHM acquired by the Poller process, and returns it to the CC. Fig. 2.13 represents a data structure used in the reply message. When the CS receives a dump request of the IPC-SHMs from the CC, it dumps all the data in the IPC-SHM ring-buffers to the specified files. The dumped files can be utilized to diagnose the Poller processes and the CS process. When the CS receives a termination request from the CC, it terminates the Poller processes and frees the created IPC-SHMs.

The CS checks the Poller status and the latest data-collection time-stamp in the IPC-SHM because the Poller process sometimes stops due to hardware I/O condition. If the latest time-stamp is not updated within a given period, the CS regards that the Poller process is down or in some troubles. Then, the CS terminates the Poller and restarts the Poller again. This function of the CS contributes to improve availability of the data acquisition system.

Collector Client

A CC process runs on an operator console. One CC process is prepared for each equipment group such as SR magnet, SR vacuum, linac and so on. The CC works as a client process to periodically collect monitoring data from related CS processes, and puts the collected data in an on-line database. A data installation cycle is determined as a least common multiple of data-acquisition cycles of all the related Poller processes.

The CC is controlled by sending S/V/O/C messages, as shown in Fig. 2.14. When the CC receives a "*start*" message, it initializes the related CSs and the Pollers, then starts the data collection. When the CC receives a "*bye*" message, it stops the data collection and instructs the CS to terminate the Poller processes. The data collection pause by receiving a "*pause*" message from the CC, and starts again by receiving a "*resume*" message.

If a timeout occurs in a communication between a CC and a CS, and the CC fails to reconnect with the CS by specified number of times, then the CC gives up data collection from the CS and sets the fail data in the on-line database. After fixing the trouble and getting ready to restart a data collection, the CC is able to reconnect with the CS by receiving "*reconnect*" message while it pauses in the data collection.

2.2.2.7 Database

The MADOCA framework had been built fully based on a database system [38][39]. All data required for the machine operations and collected from the machines are stored in the database. A consistent data structure and common data access methods are necessary to manage a large amount of data. Hence, Sybase Adaptive Server Enterprise (ASE) has been introduced as a relational database management system (RDBMS). It provides not only a convenient way to store data but also a unified, simple and fast data access. The database is designed along the relational database methodology and normalized data tables. Three kinds of databases have been built, i.e. a parameter database, an on-line database, and an archive database.

Parameter database

A parameter database manages static part of the database. The parameter database contains tables of following categories:

- Attributes of the equipment and device information required for the data acquisition of the equipment.
- Beam parameters and machine operation parameters for the equipment.
- Calibration data for the equipment such as BPMs and magnets and so on.
- Data buffer for communication between operation programs running on the separate operator consoles. These tables are used for exclusive control of the programs.
- Alarm information of thresholds for analog signals and reference bits for digital signals [40].

According to operation conditions, the machine status such as machine optics, bunch-filling pattern and setting values for related equipment have to be changed. At machine tuning time, experts of beam dynamics look for suitable parameters of the accelerators and save them into the RUN_SET table. Operators easily can reproduce the previous machine status by loading the necessary RUN_SET data from the parameter tables.

On-line database

An on-line database stores the present status of the accelerators collected by the poller/collector data acquisition system. Since high throughput of data storing and retrieving is required for the on-line database, the table size of the online database is limited. Application programs monitoring the status of the accelerators retrieve the latest data from the on-line database instead of direct access to the EM. This scheme reduces the network traffic and the CPU loads of the VME computers.

The on-line tables are built like ring buffer. The format of one data point is 4-byte integer or 4-byte floating-point, except for integrated beam current data that is expressed by 8-byte floating-point. Each row of the on-line table contains a sequential number column and time column as keys to the indexed access.

Archive database

An archive database permanently stores the data sampled from the on-line database. The archive database has the identical structure to the on-line database, except for the length of the row that has a limit. Data reduction processes for each equipment group periodically sample the data from the on-line database to make the archive database. In addition to the periodical insertion, there are some processes that inserts the data to the archive database directly such as the alarm surveillance process, the closed orbit distortion (COD) measurement process, the bunch-by-bunch current measurement process and so on. The size of the archive database is increasing at the rate of 100GB per year, and the total size at the end of December 2004 is about 350 GB.

The archive database is available for off-line analysis and data mining. The archive processes heavily consume a lot of server-machine resources, i.e. CPUs, disk access, and network bandwidth, so the heavy loads may slow down other tasks for accelerator control. To resolve the problem, distributed database was newly built by employing Sybase Omni Connects [41]. The Omni Connects is a standard component of a part of the Sybase ASE and helps distributed database operation as a middleware. The distributed database builds proxy database table on the main server. Database users can seamlessly access the actual data on the remote database server by accessing the proxy database table. It plays a role like the NFS data exporting mechanism of the UNIX file system on the database system. Fig. 2.15 shows a structure of the proxy table.

The archive database is separated from the main database server running on the HP cluster machine, and is built on the DELL cluster machine as the remote database. As a result of performance test [41], the overhead of the proxy table was negligible small, and the remote database showed an equal or better performance than that of the main server.

Data access functions

Two data access methods were provided to access the database for application programmers. One is a set of C function libraries, and the other is a set of CGI programs for WWW browsing.

The C functions were prepared for the application programmers who built the operation GUIs, equipment control GUIs, beam-analysis software and so on. The application programs were based on UNIX, C-language, and X-Window system. Over 400 C-functions were prepared for accessing the parameter database. The C-functions hide SQL commands from the programmers. Since the structure of the on-line database and archive database is identical, the on-line and archive database can be accessed with a small number of functions without taking into account the actual data location in the database. The data can be obtained by specifying the man-readable signal name and period of the time, or the signals that belong to the equipment group can be retrieved within a given period.

A set of CGI programs written in Python script displays the table of the signals on the Web browsers as shown in Fig. 2.16. By specifying (clicking) the signal name, a graph for any data in the on-line and archive

database is dynamically drawn by the gnuplot [42] in accordance with the user's request, as shown in Fig. 2.17. The CGI programs also display the data in a text format as shown in Fig.2.18. Web browser users can download the data for analysis.

2.2.2.8 Application programs

Most of the application programs running on the operator consoles are GUI base. A commercial GUI builder, X-Mate [43], is available to build man-machine interfaces. The look&feel of the X-Mate is based on the Motif1.2, which uses X11 protocol. The X-Mate does not need window manager like the CDE (Common Desktop Environment) because it has its own window system on top of the X-library. The X-Mate provides rapid developing environment with a comfortable editor. Application programmers can make widgets in WYSWYG (what you see is what you get) without knowing the X11 programming. Equipment control sequences can be written into the call-back routines of push buttons, pull-down menus, tables and so on. The X-Mate gratefully contributed to enhance productivity of GUI programs.

Table 2.1 Present beam parameters of the SPring-8 linac with ECS. For the injection to the NewSUBARU, beam parameters for top-up operation is used, and the beam current is reduced to a half by using beam slit in a beam transport line.

	Booster Synchrotron		Top-up
Pulse Width	1 ns	40 ns	1 ns
Repetition	1 pps	1 pps	1 pps
Current	1.7 A	70 mA	660 mA
dE/E (FWHM)	0.45%	0.55%	0.32%
Energy Stability (rms)	0.02%	-	0.01%

Table 2.2 List of VME I/O boards used at the storage ring control system.

Board type	Board name	Specifications
Analog input	AVME9325-5	12-bit ADC, 5μsec/channel throughput rate 16 differential / 32 single-ended non-isolation inputs Input range; ±5V, ±10V, 0 to 10V 128K Byte RAM for data storage Trigger source; internal timer, external signal, software
Analog input	Advme2602	16-bit ADC 8 channel isolated inputs Thermo-couples and Pt100 thermal resistance can be directly input.
Digital input	AVME9421	64-bit inputs with photo isolation from VMEbus and each other 4 to 25V DC input
Digital input	HIMV-610	96-bit TTL-level inputs
Digital output	AVME9431	64-bit outputs with photo isolation from VMEbus and each other. Max. 1A sink current from up to 55V DC source.
Digital input/output	HIMV-630	96-bit TTL-level inputs/outputs
Pulse train generator	MP0351	5 axes CW/CCW outputs Max. 240Kpps output pulse rate.
GP-IB control	Advme1543	-
GP-IB control	EVME-GPIB21	-

Table 2.3 Specifications and applications of the RIO slave boards.

Type	Size	Specifications	Response time	Applications
A	3U	AI × 1 (16-bit ADC, ≤ 125msec) AO × 1 (16-bit DAC, ≤ 1msec) DI × 8 (photo coupler isolation) DO × 8 (photo coupler isolation)	0.2msec	Magnet power supplies control
B	6U	DI × 32 (photo coupler isolation) DO × 32 (photo coupler isolation)	0.2msec	Magnet power supplies control, Vacuum equipment control
C	6U	AI × 16 (12-bit ADC, ≤ 100μsec/channel, isolation between each channel)	1.1msec	Vacuum equipment control
E	6U	AI × 1 (16-bit ADC, ≤ 32msec) 8-bit DI (photo coupler isolation) 8-bit DO (photo coupler isolation)	0.2msec	COD BPM control
F	6U	AI × 4 (12-bit ADC × 4, ≤ 2.4μsec, occupation of 5 slave-boards address)	27msec	Single-path BPM control
G	6U	DI × 16 (photo coupler isolation) DO × 64 (photo coupler isolation)	0.6msec	COD BPM control, Single-path BPM control

Table 2.4 Definitions of the fail data flag and off data flag for analog and digital data types.

	Analog data (Float)	Analog data (Integer)	Status data
fail data	8.88×10^{32}	0x7fffffff	0x80000000
off data	-8.88×10^{32}	0x80000000	0x40000000

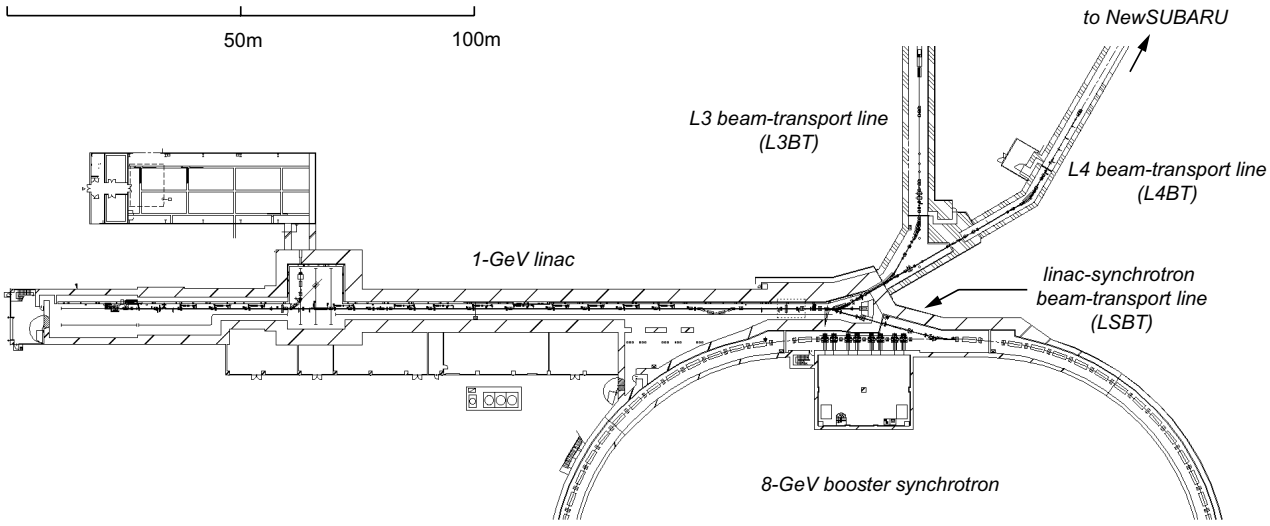


Fig. 2.1. Over view of the 1 GeV linac and three beam-transport lines; LSBT, L3BT and L4BT.

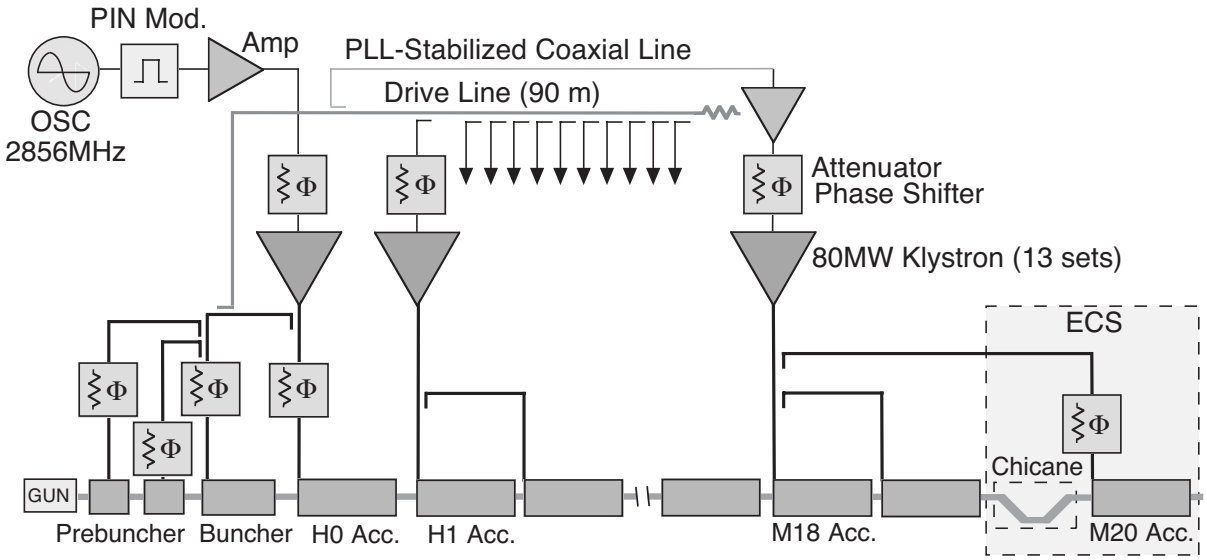


Fig. 2.2. Block diagram of the SPring-8 linac RF system.

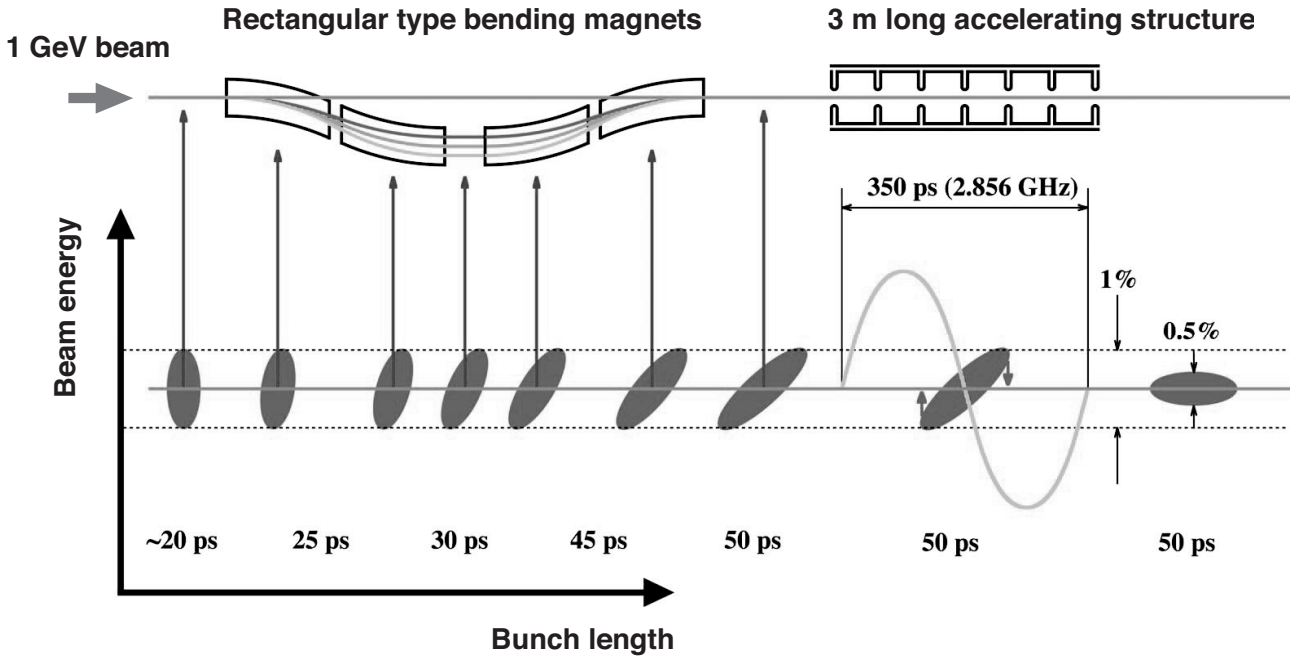


Fig. 2.3. Components of the energy compression system (ECS) and beam compression process at the ECS. The zero crossing RF power shapes the energy spread of longitudinally extended beams.

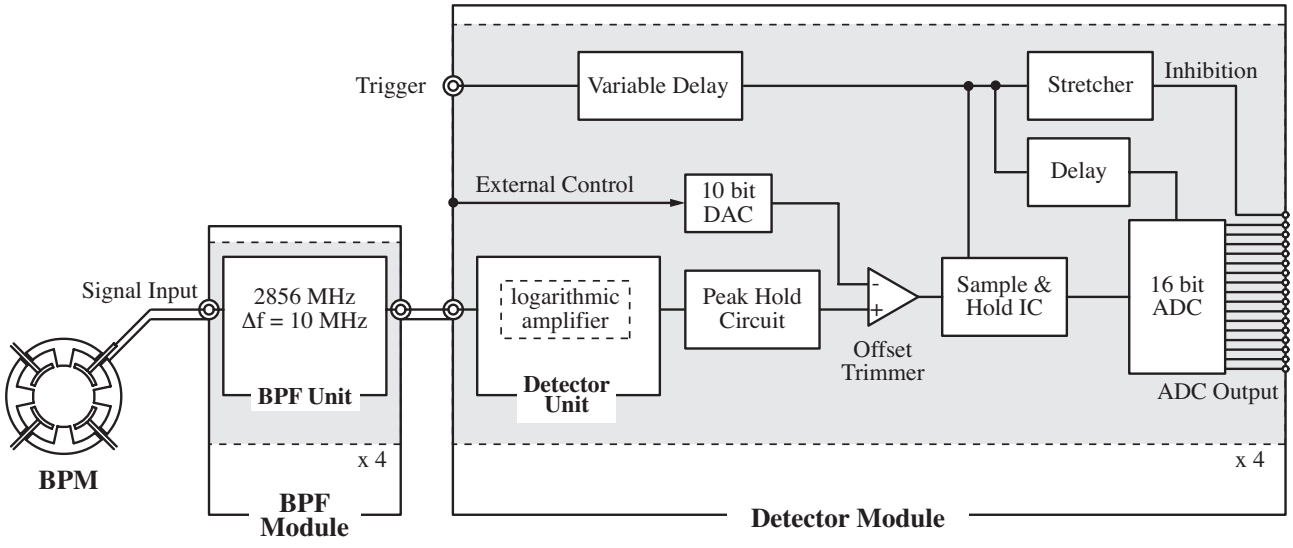


Fig. 2.4. Schematic diagram of the linac BPM system. The BPM system consists of a four-channel electrostatic strip-line monitor, a 2856 MHz BPF module, and a detector module. Output of the detector module is four sets of 16-bit TTL-level digital signals and an inhibition signal.

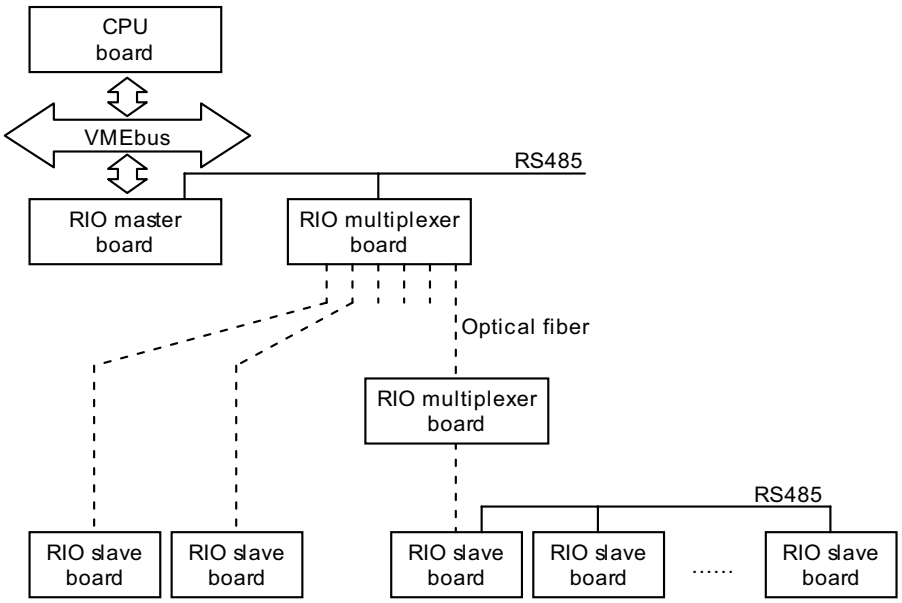


Fig. 2.5. Schematic diagram of the storage ring RIO system.

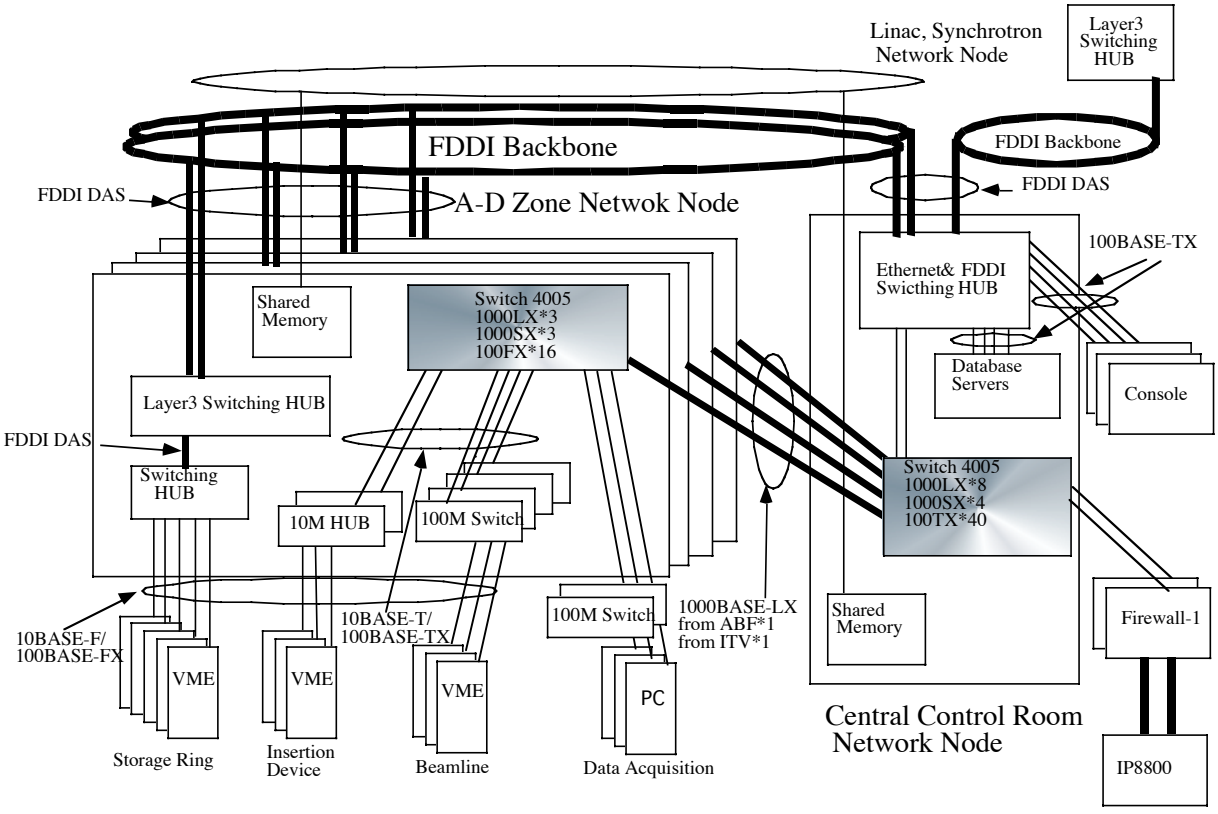


Fig. 2.6. Schematic view of the SPring-8 accelerator control and beam-line control network.

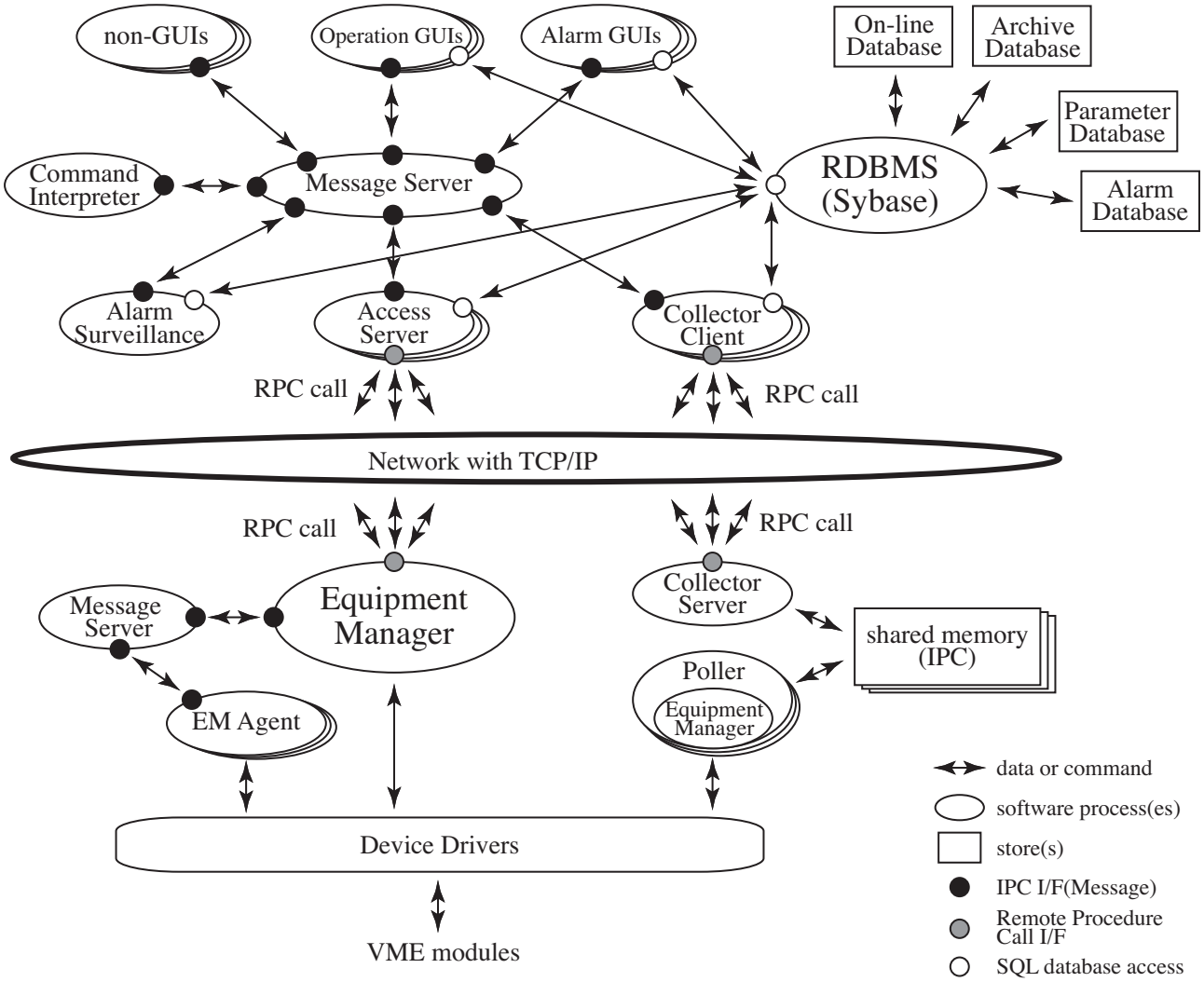


Fig. 2.7. Schematic software diagram of the SPring-8 standard control framework MADOCA (Message And Database Oriented Control Architecture).

sr_ms_serve	MS	sp8opr control oper beamd srmag srrf srvac srmon linac nsopr synchro
sr_ms_manage	MS	sp8opr control oper beamd srmag srrf srvac srmon linac nsopr synchro
sr_magtmp_cc	srmagtmpcc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mag_cc	srmagcc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mag	srmagas	sp8opr control oper beamd srmag srrf srvac srmon linac linac
sr_rf_ccg	srrfas	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_rf_cc	srrfcc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_rf	srrfas	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_vac_cc	srvaccc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_vactmp_cc	srvactmpcc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_vactmp	srvactmpas	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_vac	srvacas	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mon_dcct_cc	srmondcctcc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mon_cc	srmoncc	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mon_rfbpm_we7k	srrtmas	sp8opr control oper beamd srmag srrf srvac srmon linac
sr_mon	srmonas	sp8opr control oper beamd srmag srrf srvac srmon linac
……		

Fig. 2.8. An example of the ACL (Access Control List) file for the MS (Message Server). The first column means object name, and the second column shows responsible server name. For example, if an MS receives an S/V/O/C message of the object whose name starts from “sr_mag_cc”, the MS forwards the message to the server process “srmagcc”. The rest of the column is the list of the privileged user accounts. For example, users of sp8opr, control, oper, beamd, srmag, srrf, srvac, srmon and linac account can control “sr_mag_cc” object group.

```
V/O-1
        C-1     function_name_of_execution_process-1          arg1-1-1    arg1-1-2    …
                function_name_of_interpretation_process-1     arg1-2-1    arg1-2-2    …
                function_name_of_abstraction_process-1        arg1-3-1    arg1-3-2    …
        C-2     function_name_of_execution_process-2          arg2-1-1    arg2-1-2    …
                function_name_of_interpretation_process-2     arg2-2-1    arg2-2-2    …
                function_name_of_abstraction_process-2        arg2-3-1    arg2-3-2    …
V/O-2
        C-3     function_name_of_execution_process-3          arg3-1-1    arg3-2-2    …
                function_name_of_interpretation_process-3     arg3-2-1    arg3-2-2    …
                function_name_of_abstraction_process-3        arg3-3-1    arg3-3-2    …
……
```

Fig. 2.9. Basic format of the *config.tbl*. Function names and arguments are written in ASCII format files.

```
#
put/sr_mag_ps_st_v_1_1
  on            em_mag_st_on            /dev/rio_0      1
                none
                em_std_ret
  off           em_mag_st_off           /dev/rio_0      1
                none
                em_std_ret
#
set/sr_mag_ps_st_v_1_1
   %fA          em_mag_st_current_put   /dev/rio_0      1
                em_mag_st_conv_put      1       6.5535e+3       3.27675e+4
                em_std_ret
#
get/sr_mag_ps_st_v_1_1
   status       em_mag_st_status_get    /dev/rio_0      1
                none
                em_mag_st_status_ret
   current_adc  em_mag_st_adc           /dev/rio_0      1
                none
                em_mag_st_conv_get      1       1.57168e-4      -5.15
   current_dac  em_mag_st_dac           /dev/rio_0      1
                none
                em_mag_st_conv_get      1       1.525902e-4     -5.0
……
```

Fig. 2.10. An example of a *config.tbl* file. This example is a part of the config.tbl for magnet power supplies control of the storage ring. In this example, there are three V/O sets, and total six combinations of V and O/C are shown.

<collectorserver>id=1,name=srmagacs,inittry=10,memdumpdir=/home/sr/control,diagdumpdir=/home/sr/control,maxproc=5</collectorserver>
<poller>id=1,cycle=2.0,name=srmagapl1,table=/prj/bin/magps_a/poller_fast/config.tbl,ringsize=10,exec=/prj/bin/magps_a/poller_fast/pc_po_main,serverid=1,inittimeout=120,interval_n=3,offset_t=10,maxretry=1,diagsize=5</poller>
<signal>id=10001,pollerid=1,type=float,kind=1,signame=sr_mag_ps_b/current_dac</signal>
<signal>id=10002,pollerid=1,type=float,kind=1,signame=sr_mag_ps_b/current_adc</signal>
<signal>id=10003,pollerid=1,type=int,kind=2,signame=sr_mag_ps_b/status</signal>
<signal>id=10004,pollerid=1,type=float,kind=1,signame=sr_mag_ps_q_main_1/current_dac</signal>
<signal>id=10014,pollerid=1,type=float,kind=1,signame=sr_mag_ps_q_main_1/current_adc</signal>
<signal>id=10024,pollerid=1,type=int,kind=2,signame=sr_mag_ps_q_main_1/status</signal>
<signal>id=10005,pollerid=1,type=float,kind=1,signame=sr_mag_ps_q_main_2/current_dac</signal>
<signal>id=10015,pollerid=1,type=float,kind=1,signame=sr_mag_ps_q_main_2/current_adc</signal>
<signal>id=10025,pollerid=1,type=int,kind=2,signame=sr_mag_ps_q_main_2/status</signal>
……

Fig. 2.11. An example of a PCMF (Poller/Collector Management File). It contains information related to the CS and the Pollers such as the executable filenames of the Pollers, polling cycles, and read signals.

<table>
<tr><td rowspan="10">Header part</td><td colspan="6">Poller ID</td></tr>
<tr><td colspan="6">Poller status</td></tr>
<tr><td colspan="6">Total number of records</td></tr>
<tr><td colspan="6">A number of signals per one record</td></tr>
<tr><td colspan="6">A record number of the newest acquisition</td></tr>
<tr><td colspan="6">Time of the newest acquisition</td></tr>
<tr><td colspan="6">Restart time</td></tr>
<tr><td colspan="6">A number of restart times</td></tr>
<tr><td colspan="6">Reserve area (256byte)</td></tr>
<tr><td rowspan="12">Record part</td><td rowspan="3">Record number 1</td><td rowspan="3">Acquisition time</td><td rowspan="3">Acquired data array</td><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td rowspan="3">Record number 2</td><td rowspan="3">Acquisition time</td><td rowspan="3">Acquired data array</td><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td>…</td><td>…</td><td colspan="4">…</td></tr>
<tr><td rowspan="3">Record number N</td><td rowspan="3">Acquisition time</td><td rowspan="3">Acquired data array</td><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
</table>

Fig. 2.12. A data table structure of the poller is shown. A table is built on the shared memory.

<table>
<tr><td colspan="4">Error status of the CS</td></tr>
<tr><td colspan="4">Poller ID</td></tr>
<tr><td colspan="4">A number of signals in one record</td></tr>
<tr><td colspan="4">Record number of the replied data</td></tr>
<tr><td colspan="4">Acquisition time</td></tr>
<tr><td colspan="4">Size of the acquired data array</td></tr>
<tr><td rowspan="3">Array of collected data</td><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>Signal ID</td><td>Data type</td><td>Data</td></tr>
</table>

Fig. 2.13. A data structure used in the data-collection reply message from a CS to a CC.

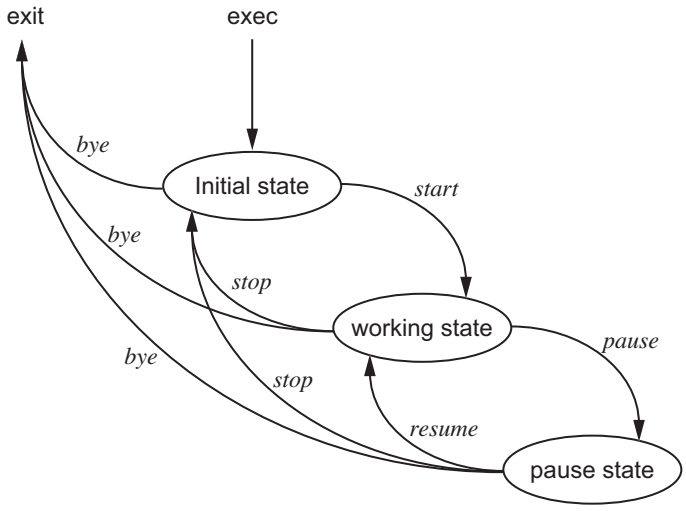


Fig. 2.14. Control message flow for the CC and transition states of the CC.

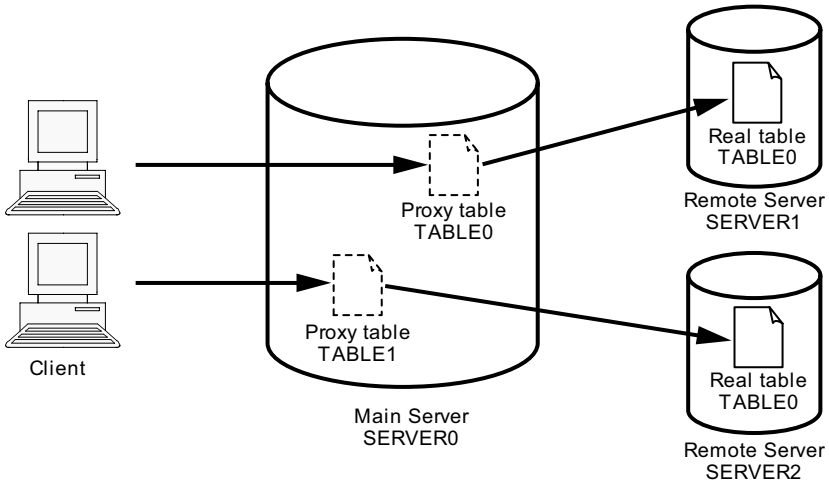


Fig. 2.15. Schematic view of a proxy database table used in a distributed database technology.

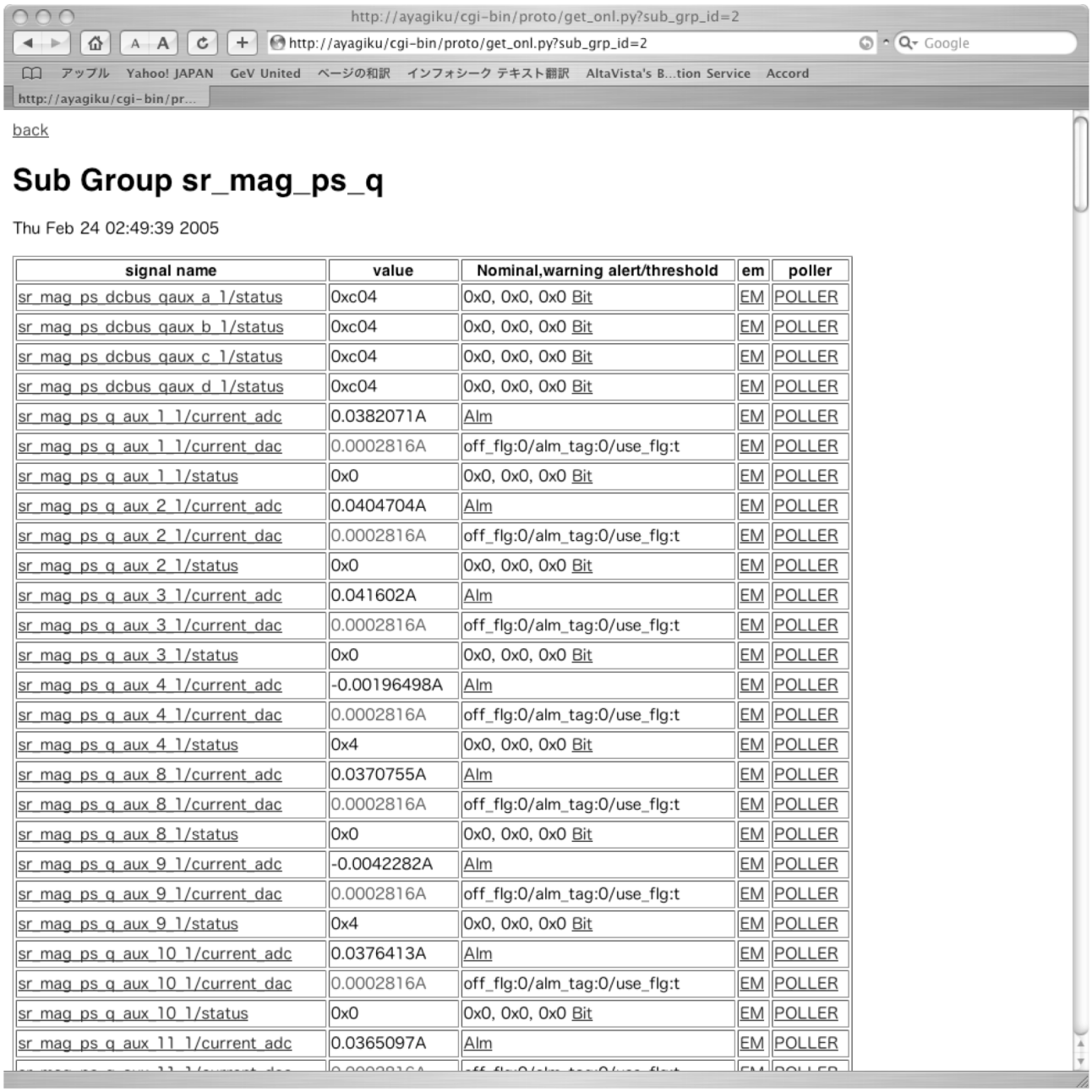


Fig. 2.16. An example of a Web page of signal table displayed using CGI programs written in Python.

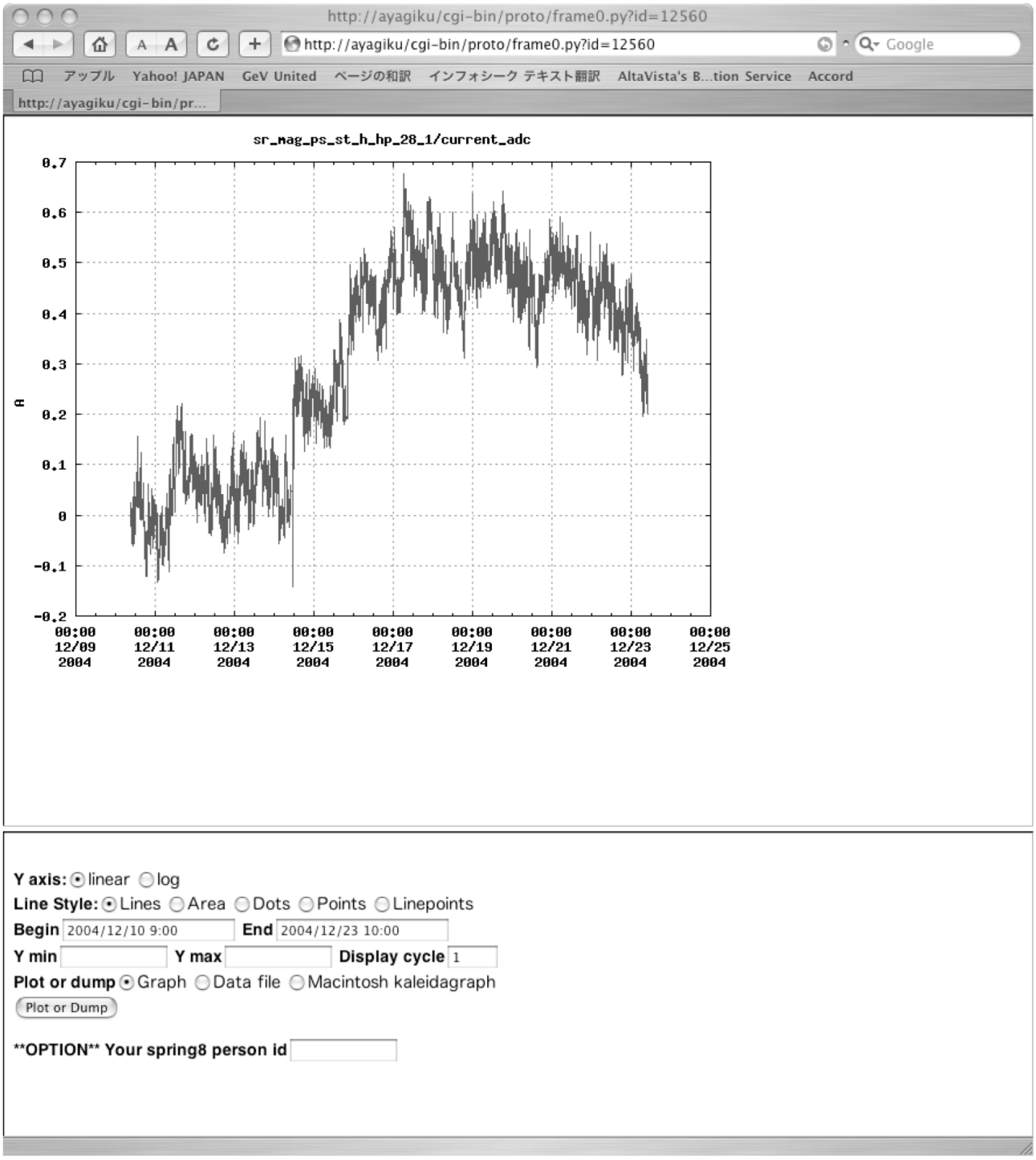


Fig. 2.17. An example of a graph of a power supply data drawn by the gnuplot. The beginning time and the end time of a graph can be chosen.

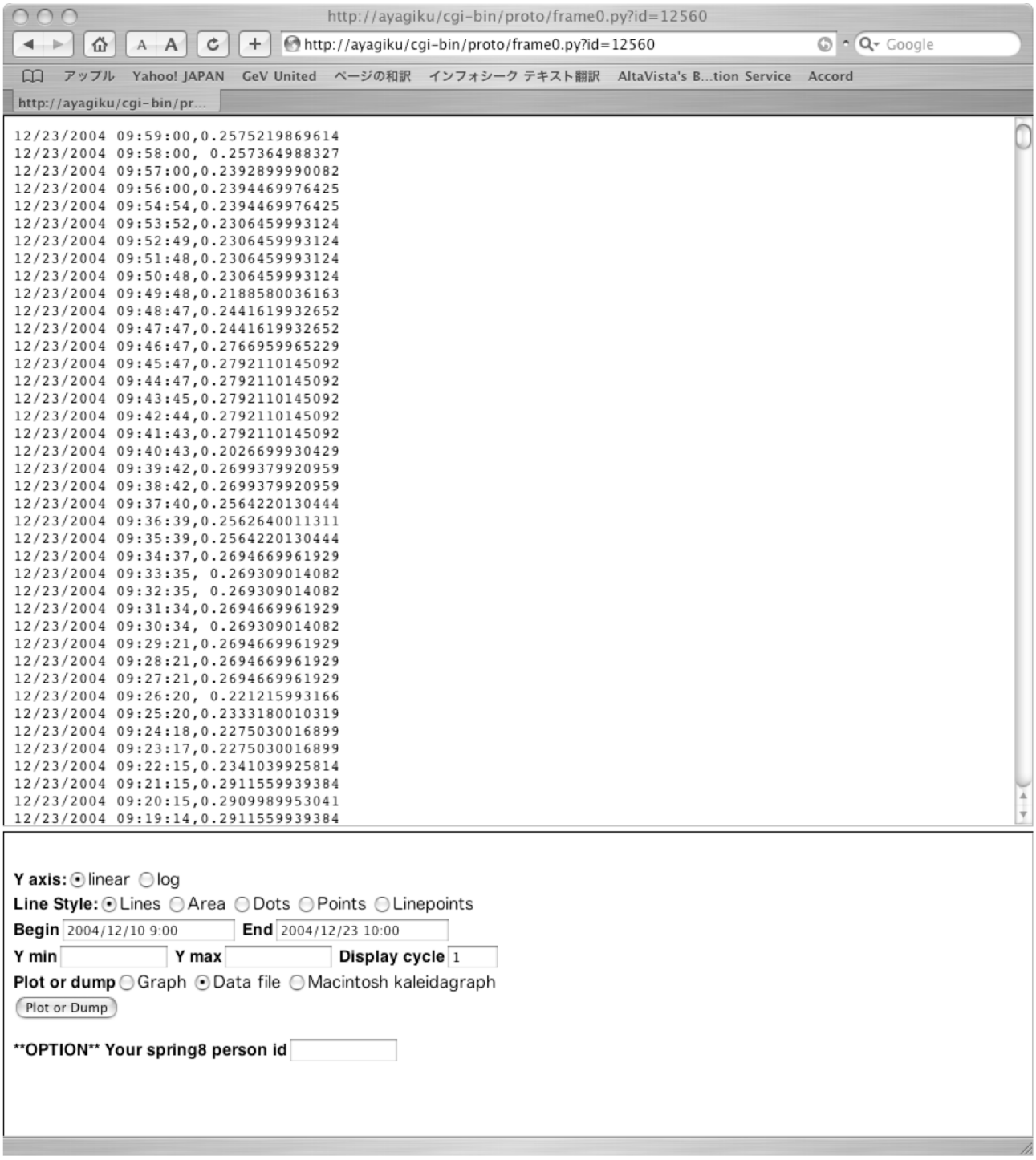


Fig. 2.18. An example of data displayed in a text format. Same as the graph, the beginning time and the end time can be chosen. The raw data can be saved to an ASCII file in the text format for analyses.

3. Development of the SPring-8 linac control system

3.1. Background of the development

The old control system was originally designed in 1991 [44]. A total number of 25 VME systems were installed along the length of the linac and the three beam-transport lines. MVME 147SA-1 [45] CPU board, equipped with a 33MHz 68030 CPU and OS-9 [46] operating system, was used as a VME controller. Each VME system equipped with five kinds of I/O boards, i.e. analog input (AI) boards, analog output (AO) boards, digital input (DI) boards, digital output (DO) boards, and pulse-motor control (PMC) boards. Table 3.1 shows specifications of VME I/O boards. Software of the old control system had its own control software differed from MADOCA. Operation software panels were developed with Motif on HP-UX. The operation panels were running on a control server workstation and were displayed on several X-terminals as operator consoles. A non-standard special protocol was developed and used for the communication between VME systems and the control server workstation. Fig. 3.1 shows the schematic view of the equipment level of the old control system.

As already described in the section 1.1, lack of capability of the old control system was the problem toward the realization of the stable top-up operation.

Software of the old control system didn't have enough stability. The software troubles sometimes led to restart or reboot of the front-end VME computers and computing system for man-machine interfaces. And a round-trip time of issued control message between an operator console and a front-end controller took one second or more, so that the old control software didn't provide good controllability for the machine operations. Moreover, the old software didn't have data logging system to help the investigation of the electron beam instabilities using the accumulated data such as the equipment status and environmental data.

About the hardware, there occurred some troubles with the front-end VME computers such as unexpected reboots, breakdowns of CPU boards and AI boards. In addition, accuracy of the acquired AI signal data was insufficient. It was suspected that these instabilities of the control hardware resulted from noises. Since the I/O boards were directly interfaced with the linac equipment, the VME systems had to be placed closer to RF klystron modulators for example, and suffered big electric noise from modulators. Moreover, any reboots of the front-end controllers by the troubles of either the hardware or the software interrupted the linac operations because the AO boards stopped the outputs of the signals and the PMC boards lost the origin of an encoder.

It was quite essential to develop a new linac control system to have enough control capabilities required for the top-up operation.

3.2. Methods of the development

The MADOCA framework was suitable for the enhancement of the control software capability. Radical re-engineering through the new development of appropriate component technologies was essential for the enhancement of the control hardware capability.

MADOCA was the SPring-8 standard software framework already had used in the control system of the SPring-8 storage ring, the booster and the NewSUBARU storage ring. The framework successfully brought stable operations together with sufficient control capabilities to these accelerators. The linac control system would have stability and a powerful data logging system, which provide off-line analyses capabilities, by introducing MADOCA. On the other hand, the MADOCA framework was not ready for a sophisticated shot-by-shot data acquisition system because the framework had been originally developed for static accelerators such as the SPring-8 storage ring. Developing a shot-by-shot data acquisition system as the standard framework of MADOCA was quite essential for single-pass accelerators such as the SPring-8 linac and the coming SCSS linac to investigate beam dynamics features. Development of the shot-by-shot data acquisition system for the SPring-8 linac BPMs will be described in chapter 4.

The old VME controllers were not compatible to apply the MADOCA framework. The OS-9 was a well-designed real-time OS, but the OS-9 was slightly old-fashioned, and did not have compatibility with the SystemV UNIX on which the EM framework was developed as the base. From the viewpoint of easy porting of MADOCA, new linac control system should employ new OS with SystemV UNIX compatibility instead of OS-9 as VME controllers. The HP-RT OS had been the first candidate because it had used in the control systems of the storage-ring, the booster and the NewSUBARU. However, an HP9000/743rt CPU board, which was a unique target for the HP-RT, had been discontinued in 1999. In order to form a new VME controller, a wide-ranging and profound survey of a new OS and a new CPU board started by focusing on satisfactory real-time feature of the OS.

For the stabilization of the linac control hardware, an optical-linked remote I/O system and a network connectable pulse-motor controller were newly developed using state-of-the-art technologies and original ideas in order to keep the VME computers away from the electric noise sources and to isolate the VME computers from the I/O parts. For precise measurement of the I/O signals from the equipment, optimization of signal conditioning and re-arrangement of I/O signal cables were required. In addition, the optical-linked remote I/O system was expected to provide the capability of the control system re-arrangement in a function-intensive structure for effective equipment controls. Fig. 3.2 shows a schematic view of the equipment level of the new linac control system.

3.3. Development of component technologies for the linac control system

3.3.1. Studies of the VMEbus CPU board

A new CPU board for the new linac control system was widely surveyed by using the following criteria:

- CPU architecture must be designed with Intel-Architecture (IA-32) or IA-32 compatible, because supplies from multi vendors were expected.
- A CPU board support a widely used OS with open source policy or a free license if possible.
- Minimum running cost with easy maintenance and long lifetime is preferable.

Additionally, the board was required to satisfy the following features in order to enhance system reliability and wider choice:

- VMEbus single-slot (if possible) or at most a double-slot with a storage unit.
- A flash disk as a boot device, and passive heat sink for a CPU.
- Smaller power consumption to avoid overheating.
- A de-fact standard Universe II chip for PCI-VME bus-bridge.

As a result of the extensive survey, a XVME658 [47] CPU board was finally chosen in 1999 [48]. The XVME-658 was a single-slot VME board with an additional board for a storage device and a floppy drive, and employed a 333MHz K6-2 [49] CPU with passive heat sink, as shown in Fig. 3.3. An IDE flash disk was chosen as a storage device. Table 3.2 shows specifications of the XVME-658 CPU board. The selection of the CPU board strongly depended on a choice of newly introduced OS described in the next section.

3.3.2. Studies of the operating system for the VMEbus CPU board

For newly introduced OS, extensive studies were performed focusing on the following features; the MADOCA framework portability, real-time performance, scheduling policy, dynamic priority control, high precision timer, system management easiness, a license form, variety of support platform and so on.

It was important that the new OS had to have compatibility with the SystemV UNIX, because the MADOCA framework was using the SystemV UNIX function calls. It was also essential that the new OS was able to provide a rigid priority control of software processes, because deterministic system controllability was highly required under the client/server multi-tasking architecture. A hard real-time feature such as the absolute deterministic operability was preferable but not indispensable for the SPring-8 accelerator controls. Task switching latency and interrupt response time of candidate OSs were studied by using of several candidate CPU boards, including an XVME-658 [50].

As a result of the studies, x86 Solaris 7 was finally chosen as the OS for the linac VME systems [48]. Solaris 7 was a modern, well refined and license free OS. It was a SystemV compatible real-time OS, which

had a pre-emptive and multi-threaded kernel. The Solaris scheduler determines software process scheduling based on global priority levels as shown in Fig. 3.4. When a higher global priority process becomes ready to execute, a current process is pre-empted and the higher process is executed immediately.

Solaris 7 provides three scheduling classes such as a system (SYS) class, a real-time (RT) class and a time-sharing (TS) class. These classes have a hierarchy of global priorities as RT>SYS>TS. It indicates that a software process running at the RT class is scheduled with higher fixed priority than the SYS class. Therefore, any processes cannot pre-empt the RT-class processes except by interrupts. In other words, there are difficulties to manage a Solaris 7 system including RT-class processes. In fact, when an RT-class process fell into infinite loop sequence and went out of control, the Solaris 7 system would hang up.

At the TS class, the system scheduler dynamically changes the process priority in order to maximize system throughput. Usual software process at the TS class is not executed deterministically. On the other hand, the priority relation between two processes can be determined by controlling a user-defined priority, because a process priority at the TS class is determined by,

process priority = (scheduler-given priority) + (user-defined priority),
where

$0 \leq$ (process priority) ≤ 59,

$0 \leq$ (scheduler-given priority) ≤ 59,

$-60 \leq$ (user-defined priority) ≤ 60.

For example, we can assign a user-defined priority value of +60 to a process A, and a value of -10 to a process B, respectively. In this case, the process A always runs at the global priority of +59, while a global priority of the process B changes between 0 and +49. This means that the process A can always run prior to the process B[*]. This is good enough for device sequence control in multi-task operation, so the TS class was chosen and applied to the linac control system by considering good system controllability.

A high precision timer was required because a timer of 1msec was already used in the linac operation programs. Solaris 7 provides the high precision timer by changing HZ value of the system parameter. The default HZ value of the Solaris is set to 100, which was resulted in 10 msec resolution, like many other UNIX operating systems. But in the linac control system, the HZ value was set to 1000 in order to obtain a timer resolution of 1 msec.

A task-switching response time was measured in the Solaris 7 with 1000Hz ticks on the XVME-658 CPU board. The same measurement was performed for comparison by using an HP9000/743rt CPU board with HP-RT v2.21 OS, which is used in the storage ring control system as described in the section 2.2.1.1. As already mentioned in the previous section, the XVME-658 was equipped with a 333MHz K6-2, and the

[*] From Solaris 9, fixed-priority scheduling has been newly introduced, which varies from 0 to +60 in the global priority.

HP9000/743rt CPU board has a 64MHz PA-RISC7100LC CPU. A test program measured differences between time stamps before and after 10,000 times calls of a *usleep()* function of 10msec sleep time. The time exceeding 10msec was considered as the task-switching response time. The program of the measuring process is shown in Fig. 3.5. Two load processes were prepared for further measurement. One load process executed a shell command, "*ls –l /usr > /dev/null*", in order to generate I/O heavy loads to the system. The other process was a program to execute a *while(1)* function to generate CPU-heavy loads to the system. The task switching response time was measured by changing a priority of the test process running together with two loads. The load processes were executed as normal TS class processes. Table 3.3 shows the measurements results.

The RT class of Solaris 7 showed good real-time performance. The result of measurement (c) was most affected by the load processes, because the global priority of the measurement process was not possible to exceed +49. The measurement (b) showed better performance than (a) with respect to the worst response time, because the measurement process was always possible to run at the global priority +59 which has highest probability not to be affected by other TS class processes. The results indicate expected characteristics from scheduling policy and priority control of Solaris.

3.3.3. Development of the optical-linked remote I/O system

3.3.3.1. System design

In order to keep VME computers away from noise source such as modulators and to isolate VMEbus from I/O parts, a new optical-linked remote I/O system, *OPT-VME* [51], has been developed for the new linac control system, instead of direct I/O boards on the VMEbus [52][53]. The OPT-VME system was designed to have following features.

First, the OPT-VME system provides a sufficient real-time communication capability between a VME master board and a remote board for faster control and data monitoring. The target communication speed was 100 msec. This feature was strongly required by the linac BPM readout, which will be discussed in a chapter 4.

Second, the OPT-VME system can read-back an actual analog and digital output data of the remote boards through the master board. Even if communication troubles occur in the OPT-VME system, this feature is very useful to resolve the problems. Practically, it is difficult to resolve communication troubles occurred in the RIO system in the SPring-8 storage ring control system because the RIO system lacks this function.

Thirdly, the remote boards hold their analog and digital outputs even though the master VME system reboots or is down. This means that any reboots of the VME system don't interrupt the linac operation.

Fourthly, the OPT-VME system provides feasibility of the function-intensive re-arrangement of the linac control system. This structure is especially important for faster control of magnet power supplies. A dedicated remote board suitable for magnet power supplies control is newly designed as described in the next section.

Fifthly, the OPT-VME boards are equipped with field-programmable gate array (FPGA) to implement most of the control logic for the board in particular for the serial optical-link communication. Employing an FPGA is attractive because of its flexibility in modifying the control logic for added new functions and to upgrade the board capabilities.

Sixthly, all the remote boards have a switch for debug mode operation in which the remote board can be accessed via VMEbus. This mode is useful to check a remote board function directly on a VMEbus chassis without communication through a master board. Furthermore, the remote boards can be used as direct I/O boards on the VMEbus.

3.3.3.2. Master and remote boards

The OPT-VME system consists of master VME boards (HIMV-658A) and several types of remote boards. Currently, five kinds of remote boards have been developed, which are combo boards (HIMV-731), DI/DO boards (HIMV-616), AI-16 boards (HIMV-714), pattern AO boards (HIMV-724) and TTL-level DI boards (HIMV-615). Specifications of the OPT-VME remote boards are described in Table 3.4. The new linac control system employed combo boards, DI/DO boards and AI-16 boards.

HIMV-658A master board

Fig. 3.6 shows a picture and a block diagram of an HIMV-658A master board. The master board has four optical-link ports and each port links with a remote board. That is, the master board controls up to four remote boards. As an FPGA for the control logic, the ACEX EP1K100 [54] was employed. The same FPGA device is also used for all types of the remote boards.

For communication between the master and remote board, TOSLINK [55] was selected which is a commercially available bi-directional optical-link module. The carrier transmission rate is 10Mbps. An H-PCF optical fiber with a 200μm silica glass core and a 230μm hard plastic clad is used for the physical channel. The transmission distance between master board and remote can be up to 500m.

Depending on the type of remote board connected to the master board, two communication modes are supported. One is the *read-only mode*, which is employed mainly for the BPM readout; the other is the *read-write mode* for wider application throughout the SPring-8 control system. It takes 20μsec (40μsec) for each communication cycle between a master and a remote in the *read-only mode* (*read-write mode)*.

HIMV-731 combo board

As described in 3.3.3.1, the HIMV-731 combo board was developed especially for electromagnet power supply control. It has two sets of analog I/O ports and two sets of digital I/O ports. One analog I/O port consists of an AI and an AO channel with 16-bit resolution, respectively. The digital I/O port consists of 16-bit DI channel with one strobe-bit, and 16-bit DO channel. Electric ground for analog signals is completely separated from digital signal ground in order to achieve good measurement accuracy. A picture of the board and a block diagram are shown in Fig. 3.7.

An analog-to-digital (A/D) converter with 16-bit serial output, AD677 [20], was adopted for the AI channel. A signal input range is either ±5V or ±10V, and a single-ended type and a differential type can be chosen for input signals. Integrated nonlinearity is estimated to be ±3LSB, and differential nonlinearity is ±1LSB. An accuracy of the converted data is calibrated to be ±3LSB at a temperature of 25±5 degree. Free A/D running takes 20μsec per conversion.

A 16-bit AD660 [20] digital-to-analog (D/A) converter was used for AO channel. An output range can be set to ±5V and ±10V. Integrated nonlinearity is ±3LSB, and differential nonlinearity is ±1LSB. An accuracy of the converted data is ±3LSB at 25±5 degree. The data settling time is within 20μsec. One strobe bit is prepared to determine D/A conversion timing explicitly. When a 16-bit data is set to the HIMV-731 register with the strobe bit of zero, the conversion is not performed and output voltage is kept at this stage. When the strobe bit is set to one, a D/A chip starts the conversion of the pre-set data into an output voltage.

One digital I/O port consists of 16-bit optically coupled input with one strobe-bit, and 16-bit optically coupled output. The 16-bit input and the output channels are isolated with each other. The input voltage range is from 12V DC to 24V DC. The strobe bit prepared for the DI can be used for an additional input. A sink current of the DO is up to 50mA DC. The maximum voltage of the DO is 24V DC. One strobe bit is prepared for the 16-bit DO as well. When the strobe bit is set to one, the latest transferred 16-bit data on a register is output.

HIMV-616 DI/DO board

The HIMV-616 consists of four digital I/O ports. Each port has the same components with a digital I/O port of the HIMV-731 combo board. Specifications of the port are also same as those of the digital I/O port of the combo board. Fig. 3.8 shows a picture of the HIMV-615 DI/DO board and a block diagram.

HIMV-714 AI-16 board

The HIMV-714 consists of four sets of an AI port, which has four channels of an AI. An AI signal is digitized with AD677 A/D converter through a four-to-one analog multiplexer. An analog ground is

completely separated from a digital ground in the same manner as that of the combo board. The HIMV-714 has 16-bit resolution of a converted data as a specification, but the guaranteed resolution is 13-bit in fact. HIMIV-714 accepts an input range of ±10V, and a jumper is prepared to select either a single-ended input type or a differential type. Integral nonlinearity is ±2LSB, and differential nonlinearity is ±1LSB of 13-bit. An accuracy of the converted data is calibrated to be ±2LSB at a temperature of 25±5 degree. Either free-running conversion or external triggered conversion can be chosen with a jumper pin. The conversion time is within 40μsec. An HIMV-714 and its block diagram are shown in Fig. 3.9.

3.3.4. Development of the network connectable pulse-motor controller

In the old control system, motor control and encoder boards, V-PAK601, were used to control pulse motor drivers. Since the VME controller directly controlled the V-PAK601 boards via VMEbus, fast controllability could be obtained. On the other hand, local controllability of the motor drivers mainly at maintenance time was sacrificed. Local operations of the motor drivers needed a serial terminal or a network-connected terminal. When the control system was updated, improvement of the local controllability of the motor controller was taken into account.

A network-connectable intelligent pulse-motor controller, motor control unit (MCU), was newly developed based on an industrial controller ND-MCU [51][53]. An MCU is a 4U height, 19” rack-mountable and diskless system as shown in Fig. 3.10. It is equipped with a 10Base-T/100Base-Tx Ethernet interface and three PCI slots. The CPU is 200MHz SH-4 [56] (SH7751R), and real-time OS NORTi [57] conforming to the μiTRON4.0 [58] specification is adopted. For a local operation, the MCU has a 4” LCD with touch panel on the front panel.

In order to achieve good local controllability as well as remote operation, the MCU was designed to satisfy following features. First, the MCU performs fixed sequences such as initialization, extraction and movement, and the sequence can be modified and downloaded via a network if necessary. Secondly, the MCU converts pulse counts and encoder counts into physical values, and vice versa using embedded conversion equations. And thirdly, the MCU holds both output pulse counts and encoder counts in it.

The first feature contributes to simplifying local operations as well as remote operations, and reducing the number of network communications with the MCU. Consequently, the MCU succeeds in keeping fast network controllability. Table 3.5 shows the developed control sequences embedded in the MCU. By introducing the second features, instructions using the physical values are available not only for the remote operation but also for the local operation. This function greatly contributes to improve local operability, especially in the case of using the complicated conversion equation. The third feature brings the MCU capability of seamless operations between the local and the remote controls. Since the MCU holds the

present position for each axis, the operation of the MCU can be continued without initialization even after the client computer, which communicates with the MCU, has rebooted.

For a choice of a PCI motor control board on the MCU, functions compatibility and hardware interface compatibility with the V-PAK601 board were taken into account. As the result of control test of some kinds of actual motor drivers, a PCI-7414V [59] PCI board as shown in Fig. 3.11 was selected from some commercial products. The specification of the PCI-7414V is listed in Table 3.6. The PCI-7414V can independently drive four axes with GMR (Giant Magneto Resistance) isolated interface, so that the MCU can independently drive twelve axes at maximum. In order to keep connectivity with all the kinds of present motor drivers, two kinds of interface-boxes named ND-MCUIF01 and ND-MCUIF02 were newly prepared between the motor drivers and PCI-7414V boards. ND-MCUIF01 (ND-MCUIF02) has 1U (2U) height and 19" rack-mountable shape, and interfaces with 4 (12) axes, respectively. Fig 3.12 shows the ND-MCUIF01 and ND-MCUIF02.

For the local control of the MCU, two operation modes are prepared. One is a drive mode for each axis and the other is a setup mode of the MCU.

Fig 3.13(a) and 3.13(b) show an axis-selection panel and an operation panel on the LCD in the drive mode respectively. As shown in Fig. 3.13(b), the operation panel provides some buttons named *init*, *insert*, *extract*, *nominal* and *move to*, for an execution of the fixed sequence. For example, the origin of the motor driver is simply determined by an *init* button on the LCD because the button starts the embedded *initialization* sequence. The drive status of the axis can be monitored by using the *status* button. The status panel displays output pulse counts, encoder counts, limit switches status, result of initialization and so on as shown in Fig. 3.14. Only for the local operation, the MCU provides jog control of each axis. Fast and slow control buttons are prepared in a jog control panel as shown in Fig. 3.15. Because the MCU converts the pulse counts into the physical values and vice versa, it is possible to drive the motor driver using the physical values for the local operation. Functions of local operation for the MCU are summarized in Table 3.7.

In MCU setup mode, it is possible to set up network interface, pulse rate, signal logic, etc. The prepared functions in the setup mode are listed in Table. 3.8.

The MCU is designed to work as a socket server when it is controlled via a network. The designed socket interfaces (command formats) are listed in Table 3.9. The commands are categorized in three different groups. One is a group of the commands for the fixed sequence execution embedded in the MCU like *initialization*. Second is for choosing conversion equations between the pulse counts and physical values. And third is a group of the other commands such as getting a current status of the MCU, setting an output pulse rate, etc. For the conversion equations, up to ten equations with ten arbitrary coefficients can be assigned and downloaded into the MCU in advance. For each axis, three equations, which are a conversion of pulse counts into physical values, a conversion of encoder counts into physical values and a conversion of physical values into pulse-counts, need to be chosen from the embedded equations via a network. Of course,

the coefficients for each conversion equation need to be given via a network as well. These conversion equations are useful to specify physical values to set the destinations, or indicate the current positions.

Since the operation task for each axis can be concurrently executed on the NORTi OS, up to twelve axes can be operated at the same time by either the remote control or the local control. This feature reduces the total execution time of the fixed sequences, especially of the initialization sequence.

For operation of the MCUs via a network, new software framework “Device Masquerade” was introduced [60]. This framework treats the MCU as a pseudo local device on a computer on which a socket client process is running. The network connected device can be controlled as a pseudo local device on the computer. It is named as a universal pseudo device (UPD) for general-purpose applications. Fig. 3.16 shows a schematic diagram of the Device Masquerade. Typical communication time of the MCU to get current status using the UPD via 100Mbps Ethernet was about 90msec. The most of the communication time was almost consumed inside the MCU.

3.3.5. Development of the new connector-boxes

The old control system employed connector-boxes to receive external signal cables with connectors, marshal the cables, and condition the signals. And in order to cut RF noise derived from the linac equipment such as modulators, the old connector-box was completely covered with a shield and directly attached to VME chassis, as shown in Fig. 3.17.

There were serious problems with the old connector-box and the VME chassis as follows. First, it was difficult to maintain the connector-boxes and the VME chassis. Access to the inside of the boxes was considerably difficult as shown in Fig. 3.18. Adding new signals to the VME computers through the connector boxes was not practical. Actually, some new signals had been directly connected to the VME I/O board without using the connector boxes. Secondly, it was impossible to check signals at the connector-box since the box did not have any terminals for signal probing. Especially for analog signals, such terminals were necessary when there were problems with the signals. Thirdly, conditioning for analog signals was not proper. In old control system, all the analog signals were directly connected into AI boards together. It was found that some analog signals had to be isolated from the ground of the connector-boxes. For example, analog signals from ion pumps were grounded to the accelerator tube, which had different ground level with the connector-box, so that these analog signals had to be isolated. On the other hand, analog signals from wire grid monitors (WGMs) had to be grounded on the reading side because these signals were completely isolated at the WGMs side. Moreover, many AI boards, which had high input-impedance of 5000MΩ, were broken because of the input signal over-voltage caused by pulse noise, resulting in necessity of impedance matching or of signal isolation. Fourthly, airflow in the VME chassis was not proper. The chassis had three exhaust fans on the upper part of rear panel. However, there were two big holes for signal cables between the

VME chassis and connector-boxes, so that the air from the front part of the VME chassis hardly passed through between VME boards. Since x86 CPU boards had been employed, the old VME chassis had possibility to cause a heat problem of the CPU boards. When the linac control system was renovated, connector-boxes had to be re-engineered to improve maintainability and to optimize signal conditioning [53].

Fig. 3.19 exhibits a plan of the new connector-box, and Fig. 3.20 shows appearance of the new connector-box. As shown in Fig. 3.21, the new connector-box furnishes a front door for easy access to the inside. And a rear panel, which can be opened as shown in Fig. 3.22, is used as a connector panel to attach connection cables from the linac equipment. Reserve space in the connector panel is prepared for D-Sub and BNC connectors. By opening the rear panel, new connectors and signal cables can be easily added in the box as shown in Fig. 3.22.

For the signal conditioning, isolation amplifiers are newly installed for AI signals except for magnet power supplies and WGMs. Since the magnet power supplies are placed close to the connector-boxes, common signal ground can be used. Since analog outputs of the WGM are completely isolated as described above, analog signals grounds of the WGMs have to be connected at the reading side.

For signals probing, three types of terminals are prepared in the new connector-box. One is a normal terminal used for AI and AO signals of magnet power supplies and AI signals of WGMs. It is not necessary for these signals to isolate. The normal terminals are settled on the sidewall of the new connector-box as shown in Fig. 3.21. The second is isolation-base terminal equipped with isolation amplifiers. The isolation-base terminals are used for analog signals from cold-cathode gauges (CCGs), ionization vacuum gauges (IVGs) and RF phase-detector and so on excluding the magnet power supplies and the WGMs. These isolation-base terminals are settled on the back of the front door as shown in Fig. 3.21. The other is relay-terminals for DI and DO signals except for magnet power supplies. For the limitation of rack-mountable space, a number of OPT-VME boards had to be reduced as much as possible. Since the OPT-VME DIO board has only four different common-pins per one board, the DIO board is able to connect four equipments if relay terminals are not introduced in front of the DIO board. By introducing relay terminals, the digital signals can be packed as much as possible. The relay terminals put on the inner wall and bottom of the new connector-boxes are shown in Fig. 3.21. Solid-state-relay modules are used for DI signals, and mechanical-relay modules are used for DO signals.

3.4. Software development of the linac control system

3.4.1. Software development for device control level

In order to adapt the new OS, i.e. Solaris 7, to the linac control system, Solaris device drivers for VME I/O boards were necessary. Table 3.10 shows a list of newly developed device drivers.

The Solaris device drivers have a hierarchy structure as shown in Fig. 3.23. Here, device drivers for nexus nodes such as bus controller are called *nexus drivers*. Device drivers for leaf nodes such as I/O boards are called *leaf drivers*. All the leaf nodes on the VMEbus are controlled through the bus nexus node of the PCI-to-VME bus bridge controller, Universe II. In the same way, leaf device drivers for the VME I/O boards are controlled through the nexus device drivers for the Universe II. Therefore it was quite important to develop a nexus device driver for the Universe II. Although the specification of the Solaris nexus device drivers were not open, the Universe II device driver as a nexus device driver was successfully developed from the source code of the open Solaris 7. The success of development of the Universe II nexus driver brings wide choice of IA-32 CPU boards as long as they employ the Universe II as PCI-to-VME bus-bridge chip.

3.4.2. Software development for equipment control and man-machine interface levels

Equipment experts developed application programs both the EMs running on the VME system and GUIs running on operator consoles [48]. Rearrangement of operation sequences and list up of equipment signals to make abstracted commands, i.e. S/V/O/C commands, took three man-months. Development of the EM (GUI) software took twelve (eight) man-months, respectively. Database parameter set-up and making of access functions took one man-month. The prototyping methodology was efficient for the rapid development of both EM and GUI programs. The device simulation codes were used for testing and debugging of the EM, and as a result, the program examinations by using real devices ran very smoothly.

The man-machine interface for the linac control was designed in two groups of GUI. One was a group of GUIs for machine operations. The other was a group of GUIs for maintenance of individual equipment. The former formed a part of the central control system. This categorization was same as the storage-ring control system and the booster control system.

At the integration time, the linac had over 270 setting values. A RDBMS consistently managed those values [48]. A total of 84 newly created tables in the database hold a set of parameters. A data acquisition process collected over 5.2kB of data from the linac every five seconds. The database system also kept this data in 18 tables. Though the linac control system needed over 100 new tables, the database system minimized the software effort to add them to the database with its standardized procedure.

3.5. Installation of the new linac control system

In the new linac control system, the VME computers for the equipment control were re-arranged in the function-intensive structure by using newly developed OPT-VME system and the MCU. And the number of the VME computers was successfully reduced from 25 to 10 with 7 VME chasses. The function-intensive structure brought tight and fast control of the equipment and provides efficient development of the equipment control software.

The OPT-VME system allowed keeping the VME CPU boards away from the noise source such as the modulators and locating them in good environment. Since the optical fiber isolated the VMEbus from the I/O boards, i.e. the OPT-RMT I/O boards, the VME computers could be guarded against the noise. Thus, introducing the OPT-VME system contributed to stabilizing the VME systems.

Since the OPT-VME remote boards could hold AO signals and DO signals independently of the state of the VME computers, it was possible to continue the linac operation even though the VME computers were rebooted or shut down. Because the present output data of the remote board could be read, the linac operation could be continued without loading any parameters from database after the VME systems started up. The OPT-VME system contributed to enhancing availability for the linac operation.

The MCU succeeded in achieving both good local controllability and fast remote operation. A total of 20 MCUs has been installed in the new linac control system. Since the MCU held the current pulse counts and encoder counts, the linac operation could be continued without initialization of the pulse motors axes when client computers are rebooted or powered off. Thus, the MCU contributed to enhancing availability of the linac operation.

All the connector-boxes, whose total number was 24, have been replaced into new ones in order to optimize and stabilize I/O signals. Fig. 3.24(a) represents an output monitor of a magnet power supply M4-2 QT-F acquired with the old control hardware. The noise level was about 0.73V (peak-to-peak). Fig. 3.24(b) shows an output monitor of the same magnet power supply acquired with the new control system. The noise level became small, which was less than 0.02V (peak-to-peak). In comparison with Fig. 3.24(a), it concludes that reliability enhancement of the monitor signals has been successful.

By introducing newly developed connector-boxes, new monitor signals could be easily added in the new control system. For examples, RF phase data of each modulator was successfully added. This signal will be used to feedback control of RF phase of the modulators. In addition, temperature data of the driveline and air in the klystron gallery were also added easily. After the control hardware re-engineering, new signals of 86 AI, 4 DI and 2 DO have been already added. Those additions contributed to stabilization of the linac operation through the signals monitoring.

The software update using the SPring-8 standard software framework MADOCA was successful. Introducing MADOCA contributed to stabilization and throughput enhancement of the linac control system.

The powerful data logging system supplied by the MADOCA framework provided the off-line analyses capabilities of the linac machine status by using C-functions and Web browsers. These off-line analyses using logging data practically contributed to the operation stabilization of the linac for the simultaneous top-up operation. The response time of the new control system was improved to about 20msec while it was in the order of a second in the previous system.

For the software update, Solaris and IA-32 based CPU boards for the VME computers were newly introduced. Solaris provided good portability of the EM framework, dynamic priority control of software processes and high-resolution timer same as HP-RT. Priority controls of both the EM and the pollers were achieved in the Solaris environment same as the HP-RT. By introducing IA-32 based CPU boards, the other substitutions are available for VME CPU boards and operating system for a long time.

The VME chasses were replaced with new ones which had enough cooling power for the VME CPU boards, resulting in prevention of the heat problems with the IA-32 based CPU boards with large power consumption. Thus, using the proper VME chassis also contributed the stabilization of the linac control system.

The other improvement was that the GUI could be built without knowledge of X/Motif programming. Easy development of the GUIs will facilitate an advanced control of the machine.

The re-engineering of the linac control system hardware and the upgrade of the software successfully brought stabilization to the linac control system itself and availability enhancement to the linac operation. Practically, seven troubles occurred which needed reboot of the VME computers in 2005. All of them were concerned with the UPD for the MCU control. There were no other software troubles such as restarts of the control panels which frequently occurred in the previous control system. And there were no hardware troubles such as VME boards breakdowns and unknown reboots of the VME computers sometimes occurred in the previous control system. Furthermore, the troubles occurred in the linac control system in 2005 never interrupted the simultaneous top-up operations. These facts indicate that the approaches in this study were very effective to the availability enhancement of the linac control system.

Finally, the linac software upgrade successfully brought seamless operation of the whole SPring-8 accelerators towards the top-up operation.

Table 3.1 List of VME I/O boards used at the old linac control system.

Board type	Board name	Specifications
Analog input	AVME9350	12-bit ADC, 33μsec/channel throughput rate 16 differential / 32 single-ended non-isolation inputs Input range; ±5V, ±10V, 0 to 10V Trigger source; internal timer, external signal, software
Analog output	AVME9210	12-bit DAC 8 channel independent outputs Output range; ±2.5V, ±5V, ±10V, 0 to 5V, 0 to 10V
Digital input	DVME DIN3	64-bit inputs with photo-isolation from VMEbus 8 bit per one common ground 4 to 25V DC input
Digital output	DVME DOUT3	64-bit outputs with photo-isolation from VMEbus 8 bit per one common ground Max. 1A sink current from up to 55V DC source
Pulse motor control	VPAK-601	Individual control of two axes CW/CCW or OUT/DIR outputs with photo-isolation for each axis ・ +12V DC output ・ Max. 240 Kpps output pulse rate using PCL-240AK† ・ 24-bit preset counter Incremental encoder inputs (A/B-phases) with photo-isolation for each axis ・ +12V DC input ・ Max. 150KHz input pulse rate ・ 24-bit counter +/- end-limit and the origin signal inputs with photo-isolation

Table 3.2 Specifications of a XVME658 CPU board.

CPU	AMD 333MHz K6-2
System bus frequency	66MHz
Chipset	Intel 430HX
L2 cache	512kB
Memory	256MB EDO DRAM
Ethernet	10Base-T/100Base-TX (Intel 82558) × 1ch
PCI-VME bus bridge	Tundra Universe IIB

† Nippon Pulse Motor Co.,Ltd http://www.pulsemotor.com/

Table 3.3 Measurement results of the task switching response time.

(a) Measuring process was run at normal TS class.

	TS (normal) without load processes	TS normal with load processes
Average (μsec)	9997	10011
Standard deviation (μsec)	140	450
Maximum time (μsec)	20235	42650

(b) Measuring process was run at TS class adding with +60 user-priority.

	TS (+60) without load processes	TS (+60) with load processes
Average (μsec)	9997	10011
Standard deviation (μsec)	147	449
Maximum time (μsec)	20087	36684

(c) Measuring process was run at TS class adding with -10 user-priority

	TS (-10) without load processes	TS (-10) with load processes
Average (μsec)	10001	10063
Standard deviation (μsec)	215	827
Maximum time (μsec)	20318	40008

(d) Measuring process was run at RT class

	RT (+0) without load processes	RT (+0) with load processes
Average (μsec)	9994	9993
Standard deviation (μsec)	9	39
Maximum time (μsec)	10778	11058

(e) Measuring process was run at HP-RT platform with same priority as load processes

	Without load-processes	With load processes
Average (μsec)	9995	10402
Standard deviation (μsec)	151	1867
Maximum time (μsec)	18575	20789

(f) Measuring process was run at HP-RT platform with +1 priority than load processes.

	Without load-processes	With load processes
Average (μsec)	9996	9987
Standard deviation (μsec)	151	159
Maximum time (μsec)	18592	18252

Table 3.4 List of newly developed OPT-VME remote board.

Board name	Board name	Specifications
OPT-RMT combo	HIMV-731	AI × 2 (16-bit ADC × 2, 14-bit accuracy) Input range; ±5V, ±10V Trigger source; internal timer Jumper selectable differential / single-ended inputs AO × 2 (16-bit DAC × 2) Output range; ±5V, ±10V DI × 32 Photo-coupler isolation 4 to 30V DC inputs DO × 32 Photo coupler isolation Isolated analog ground from digital ground.
OPT-RMT DI/DO	HIMV-616	DI × 64 + 4 (inhibit signal) Photo coupler isolation 4 to 30V DC input DO × 64 Photo coupler isolation
OPT-RMT AI-16	HIMV-714	AI × 16 (16-bit ADC × 4, 13-bit accuracy,) Input range; ±5V, ±10V Trigger source; internal timer, external signal Jumper selectable differential / single-ended inputs
OPT-RMT DI	HIMV-615	DI × 64 + 4 inhibit signal (no photo-coupler isolation) TTL level input
OPT-RMT pattern	HIMV-724	Arbitrary waveform output × 2 (16-bit DAC) Output range; ±1V, ±5V, ±10V Wave memory; 128k words × 4 / channel Maximum output rate; 100kHz

Table 3.5 Developed fixed sequences embedded in the MCU.

Sequence	Action
initialize	Determine the origin of an axis with a low speed drive.
extract	Drive a pulse motor at a low speed to the extraction limit of an axis.
insert	Drive a pulse motor at a low speed to the insertion limit of an axis.
nominal	Return to the origin of an axis with a trapezoid drive.
move to	Move to the specified position of an axis with a trapezoid drive.

Table 3.6 Specifications of a PCI-7414V motion control board

Number of axes	4 (individual control available)
Pulse output	CW/CCW or OUT/DIR output mode
	Max. 6.5Mpps output pulse rate using PCL-6045 or equivalent
	Number of output counts: -134217728 ~ 134217727
	GMR (Giant Magneto Resistance) isolation
	+5V DC output (differential line driver output)
Encoder input	Incremental encoder inputs (A/B/Z-phases)
	Max. 1MHz input pulse rate
	Counter length: 28bits
	High-speed photo-isolation
	+5V DC input
Other signal output	4-bit general purpose outputs
	+5V ~+48V DC output
	Max. 100mA output current
Other signal input	+/- end-limit and the origin signal inputs available
	12-bit general purpose inputs
	Photo-isolation
	+5V~+48V DC input

Table 3.7 List of local operation functions which the MCU prepares.

Button name	Function
init	Execute the *initialize* sequence embedded in the MCU
extract	Execute the *extract* sequence embedded in the MCU
insert	Execute the *insert* sequence embedded in the MCU
nominal	Execute the *nominal* sequence embedded in the MCU
move to	Execute the *move to* sequence embedded in the MCU
status	Display the current status of the MCU
jog	Move a pulse-motor with jog control

Table 3.8 List of prepared functions in a set up mode operation of the MCU. In the setup mode, it is possible to set up network interface, pulse rate, signal logic, etc.

	Function
Operation environment	• Set logic level of +/- end-limit, the origin and CW/CCW signals • Set +/- software limits • Set initial speed, drift speed, acceleration time and slowdown time of trapezoidal motion profile
Network environment	• Set IP address, subnet mask and default gateway of the MCU

Table 3.9 List of the designed socket interfaces (command formats) for the MCU.

Command	Operation	Function
phy_conv	put	Choose a conversion equation of pulse counts into physical values.
phy_conv?	get	Get a current conversion equation of pulse counts into physical values.
encphy_conv	put	Choose a conversion equation of encoder counts into physical values.
encphy_conv?	get	Get a current conversion equation of encoder counts into physical values.
pulse_conv	put	Choose a conversion equation of physical values into pulse counts.
pulse_conv?	get	Get a current conversion equation of physical values into pulse counts.
param_set	put	Set operation parameters of an initial speed, a drift speed, an acceleration time and a deceleration time.
param_set?	get	Get current operation parameters of an initial speed, a drift speed, an acceleration time and a deceleration time.
signal_env	put	Set operation parameters of logic levels of +/- end limits and the origin signal.
signal_env?	get	Get current operation parameters of logic levels of +/- end limits and the origin signal.
sl_position	put	Set operation parameters of +/- software limits.
sl_position?	get	Get current operation parameters of +/- software limits.
save	put	Save current operation parameters in a flash ROM.
exec	put	Execute the specified sequence of *initialize*, *extract*, *insert* and *nominal*.
move_to	put	Move to the specified physical position. Execute *move to* sequence.
move_to?	get	Get the specified physical position in the *move_to* command.
stop	put	Stop the current executing sequence and pulse output.
position?	get	Get the current status of the MCU such as an output pulse count, an encoder count, operation status, execution results of *initialize* sequence and other sequences and so on.
input?	get	Get a status of general-purpose input signals.
output	put	Put general-purpose output signals.
output?	get	Get a status of general-purpose output signals
remote?	get	Get a status of remote/local operation.

Table 3.10 Newly developed device drivers for the linac control system.

Device name	Function	Device driver type	VMEbus I/F	Boundary size
Universe II	Bus bridge	Nexus		
OPT-VME	Remote I/O	Leaf	A32 (0x09/0x0D) / A24 (0x39/0x3D) D32 / D16 / D08	512Byte
Advme1107A-08	SRAM	Leaf	A32 (0x09/0x0D) / A24 (0x39/0x3D) D32 / D16 / D08	1MByte
DVME DIN3	DI	Leaf	A16 (0x29/0x2D) D08	32Byte
DVME DOUT3	DO	Leaf	A16 (0x29/0x2D) D08	16Byte
AVME9350-I	AI	Leaf	A24 (0x39/0x3D) / A16 (0x29/0x2D), D16 / D08	1KByte
AVME9210	AO	Leaf	A16 (0x29/0x2D) D16 / D08	1KByte
V-PAK601	PMC	Leaf	A16 (0x29/0x2D) D08	64Byte

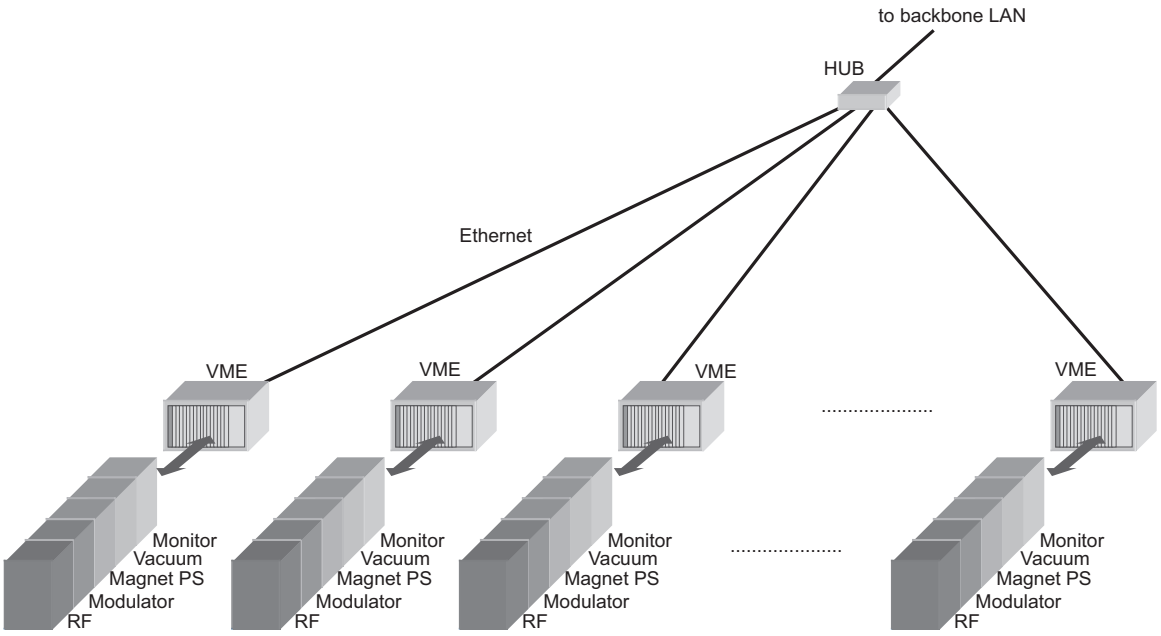


Fig. 3.1. A schematic view of the equipment level of the old linac control system. VME computers had to be placed closer to noise sources such as modulators because direct VME I/O boards were directly interfaced with the equipment. Consequently, the old control system had a place-intensive structure and required many VME computers.

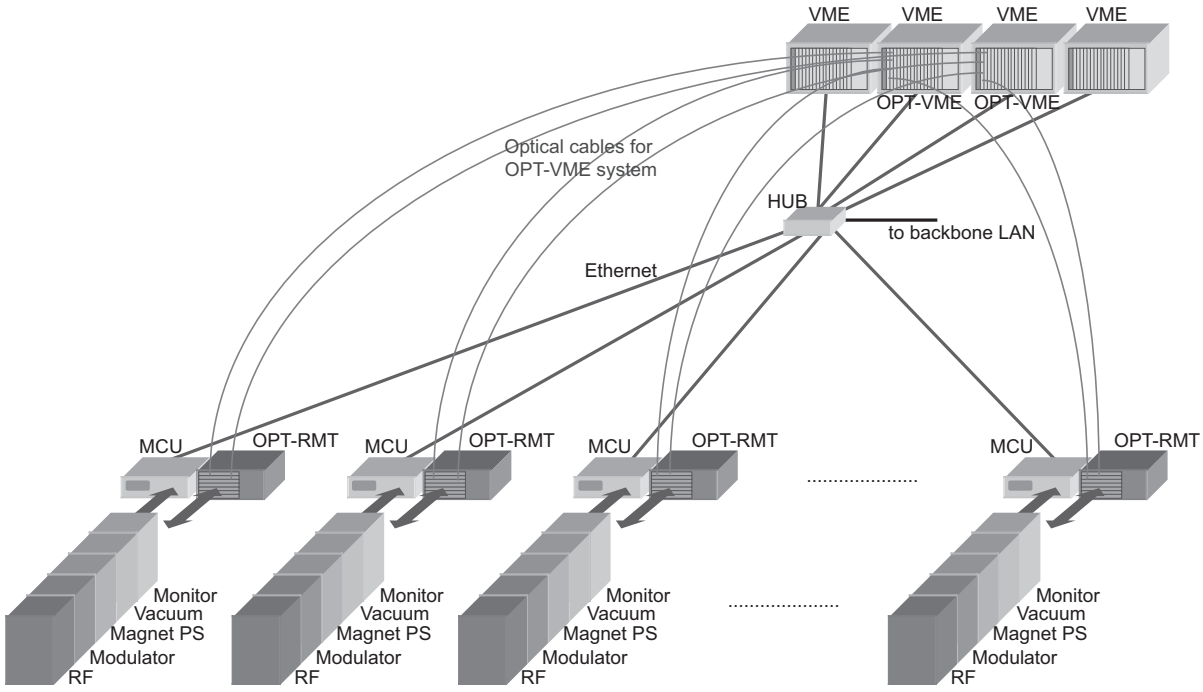


Fig. 3.2. A schematic view of the equipment level of the new linac control system. OPT-RMT boards and MCUs are deployed close to the equipment and VME computers are placed in good environment which is apart from the equipment. The VME system has a function-intensive structure by using the OPT-VME system. For example, all of the magnet power supplies for magnets in the main linac is controlled by a VME computer.

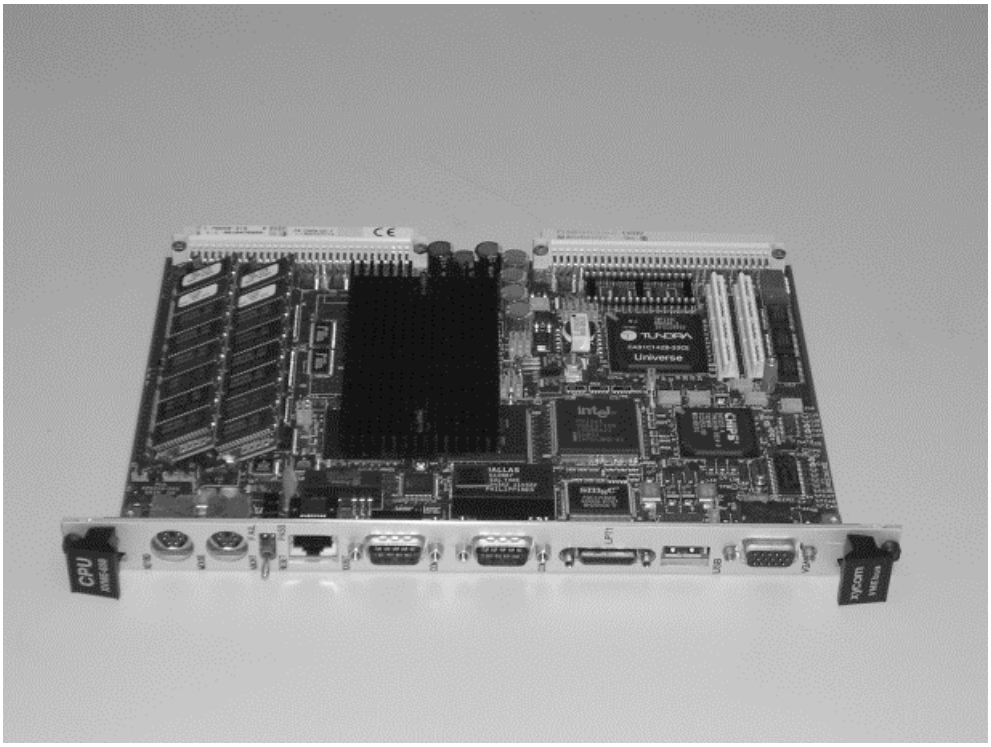


Fig. 3.3. A picture of the XVME-658 CPU board.

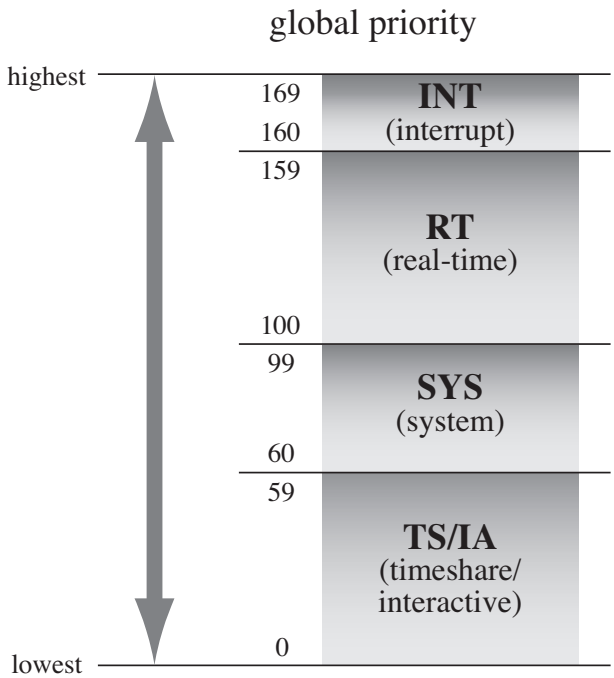
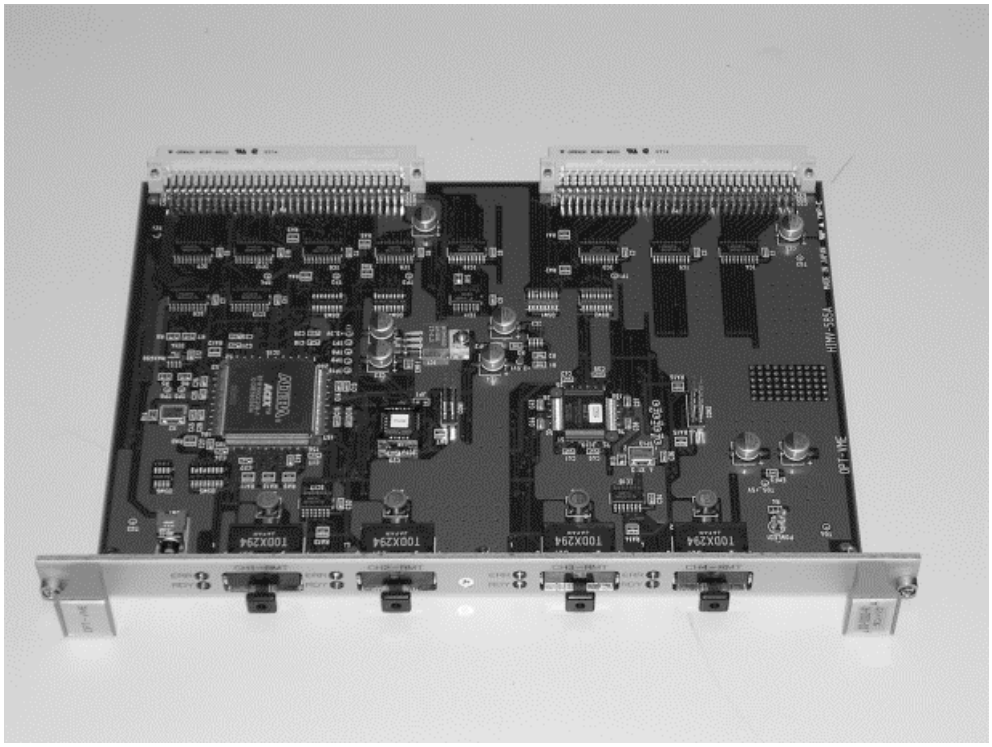


Fig. 3.4. Solaris 7 scheduling class hierarchy. Global priority of the real-time class is ranges from 100 to 159 and is above the time-sharing class (0 ~ 59) and the system class (60 ~ 99). The real-time class uses fixed priority scheduling.

```
#include <time.h>
#include <sys/times.h>
#include <unistd.h>
#include <stdlib.h>

main(int argc, char **argv)
{
      int     i;                        /* loop counter */
      int     turns, loops;
      int      cpu_time;                /* cpu time */
      struct timeval  time1[10001], time2[10001], dif;
      struct timezone zone1, zone2;
      double  uu;
      useconds_t useconds;
      struct timespec         rqtp;
      struct timespec         rmtp;

      /* argument check */
      if(argc < 2) {
            printf(" Usage :: usleeptest sleeptime(us) loops\n");
            exit(1);
      }
      /* set sleep time(micro sec) */
      useconds = (long)atoi(argv[1]);
      printf("sleep time is set to %d (us) \n",useconds);

      /* set times of test loop */
      loops = atoi(argv[2])+1;
      printf("number of test loops ==> %d\n",loops-1);

      /* start the mesurment */
      for(i=0;i<loops;i++) {
            gettimeofday(&time1[i],&zone1);
            if( usleep( useconds ) < 0 ) {
                  printf("usleep error\n");
                  break;
            }
            gettimeofday(&time2[i],&zone2);
      }

      for( i=1 ; i < loops ; i++ ) {
            dif.tv_sec=time2[i].tv_sec-time1[i].tv_sec;
            dif.tv_usec = time2[i].tv_usec-time1[i].tv_usec;
            printf("%lf\n",(dif.tv_sec*1000000.0+dif.tv_usec));
      }
}
```

Fig. 3.5. A program of the measuring process for task switching response-time.

(a)



(b)

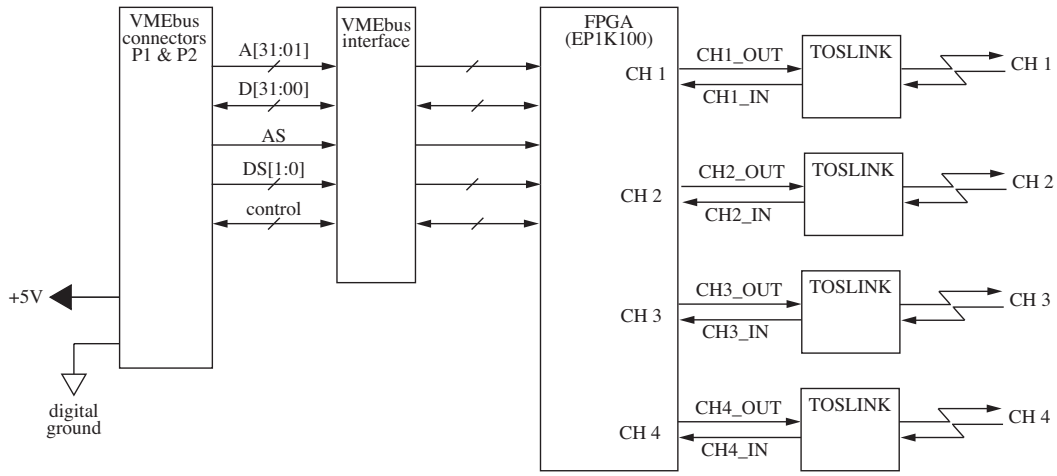
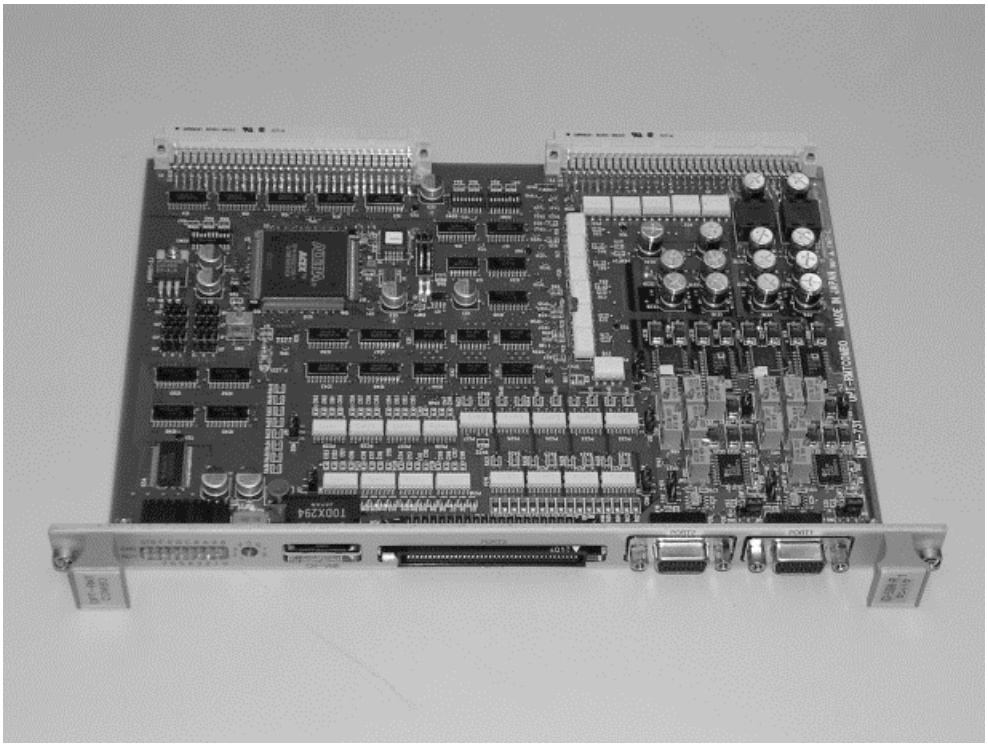


Fig. 3.6. Fig. (a) shows a picture of the HIMV-658A master board, and Fig. (b) shows a block diagram of the board. The HIMV-568A board has four optical-link connectors on the front panel, it can control four remote boards.

(a)



(b)

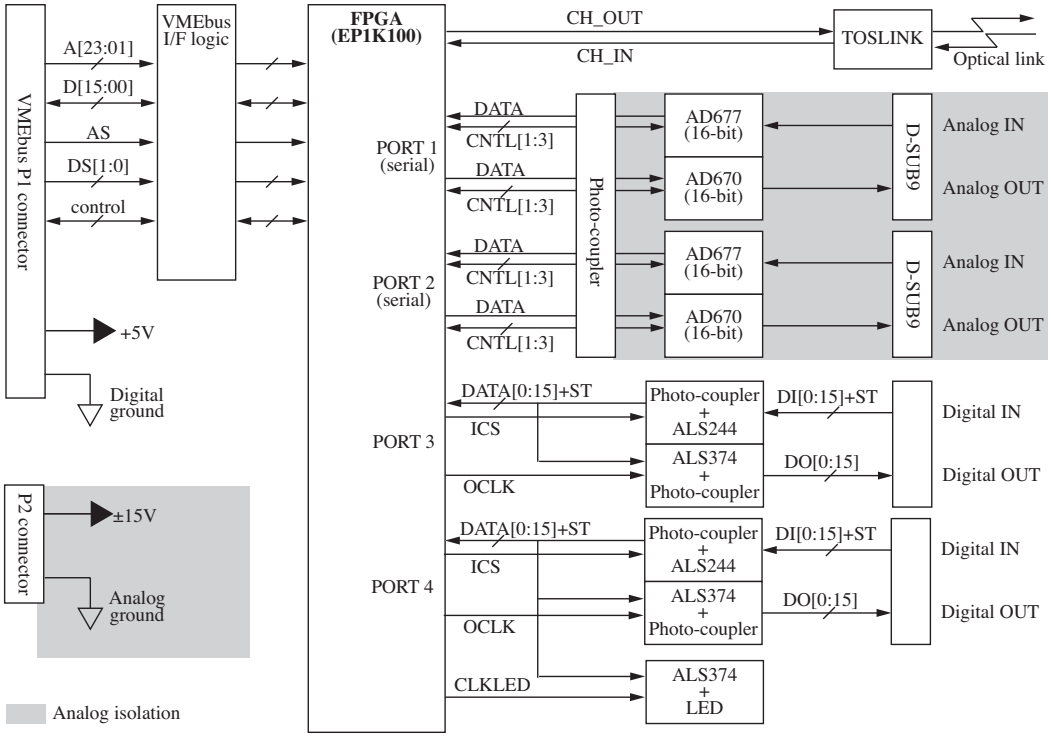
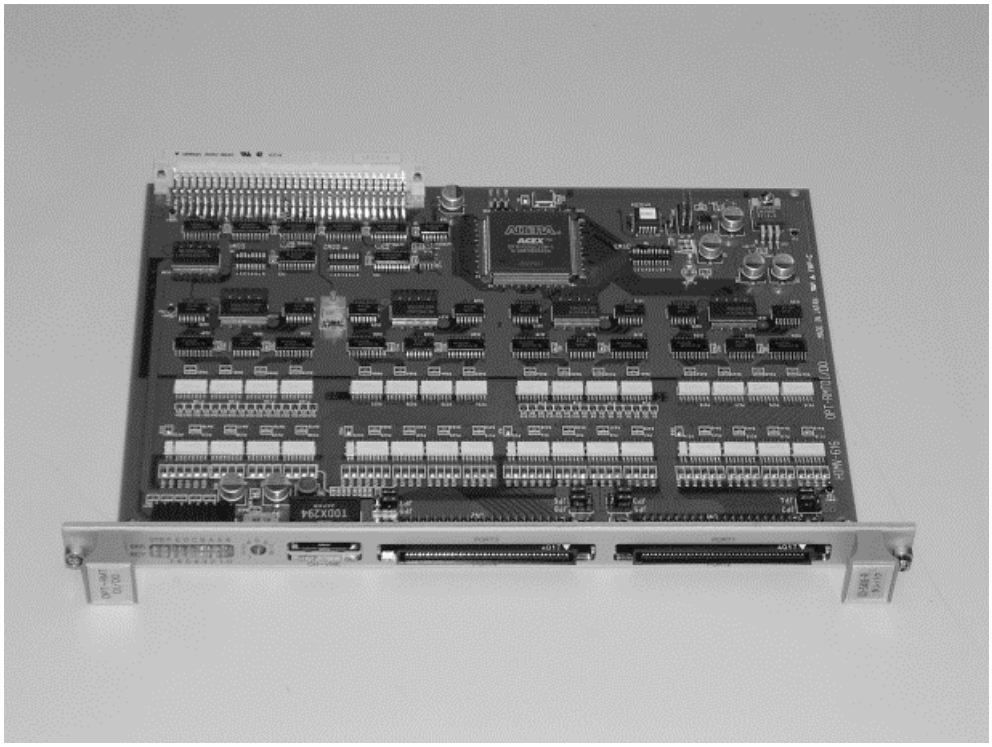


Fig. 3.7. Fig. (a) shows a picture of the HIMV-731 combo board, and Fig. (b) shows a block diagram of the board.

(a)



(b)

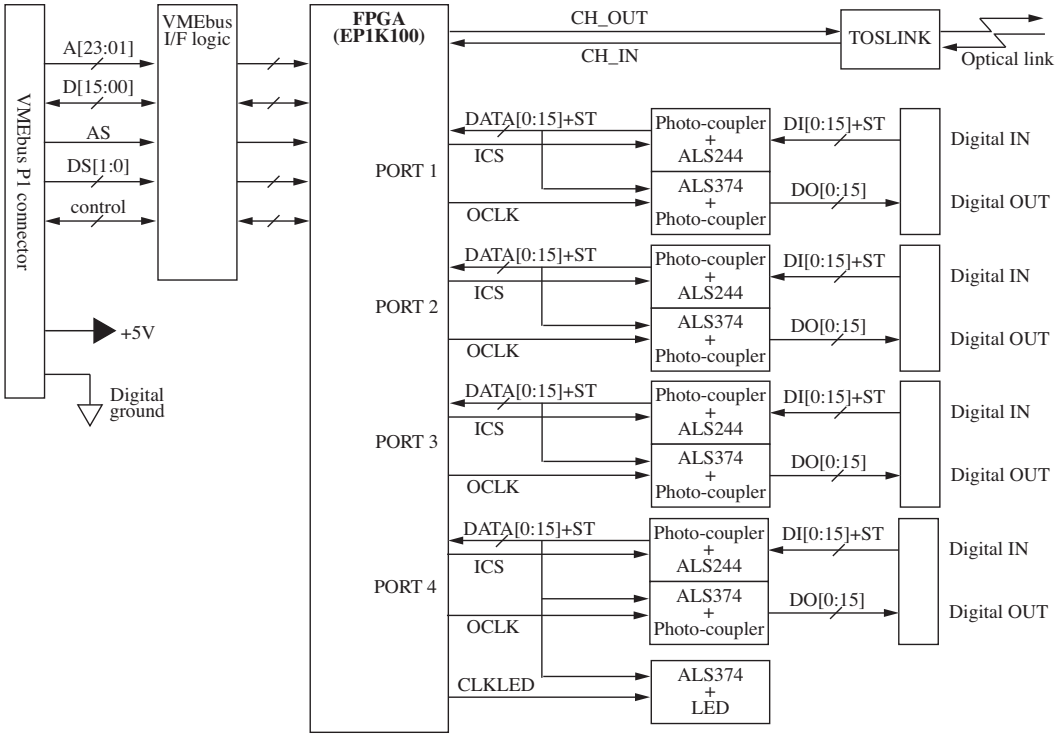
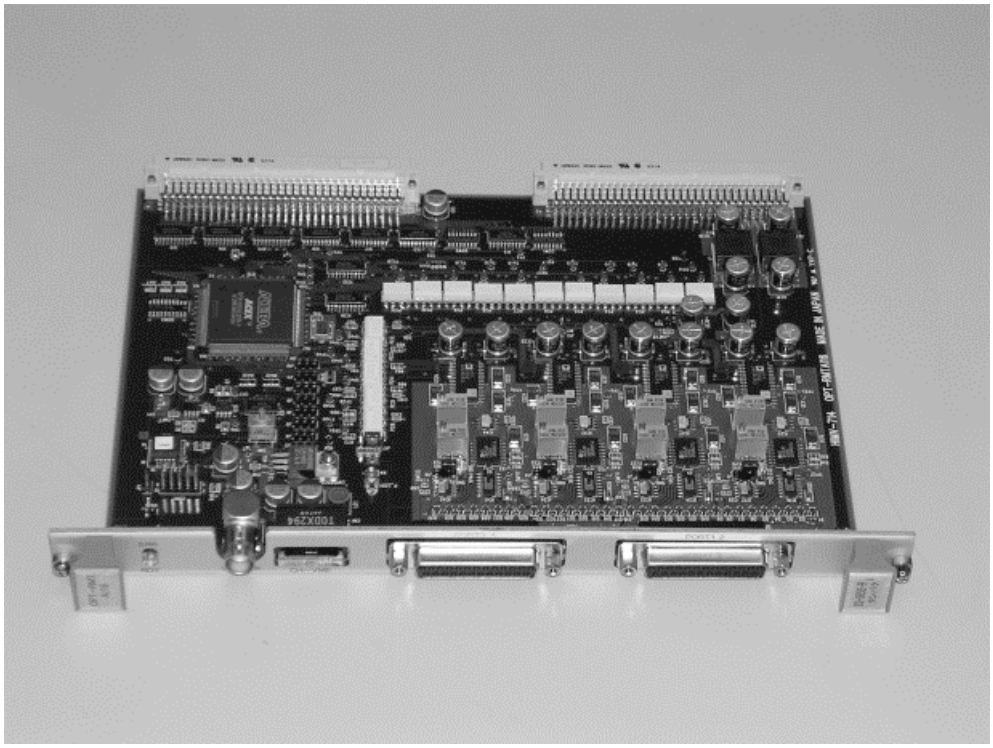


Fig. 3.8. Fig. (a) shows a picture of the HIMV-616 DI/DO board, and Fig. (b) shows a block diagram of the board.

(a)



(b)

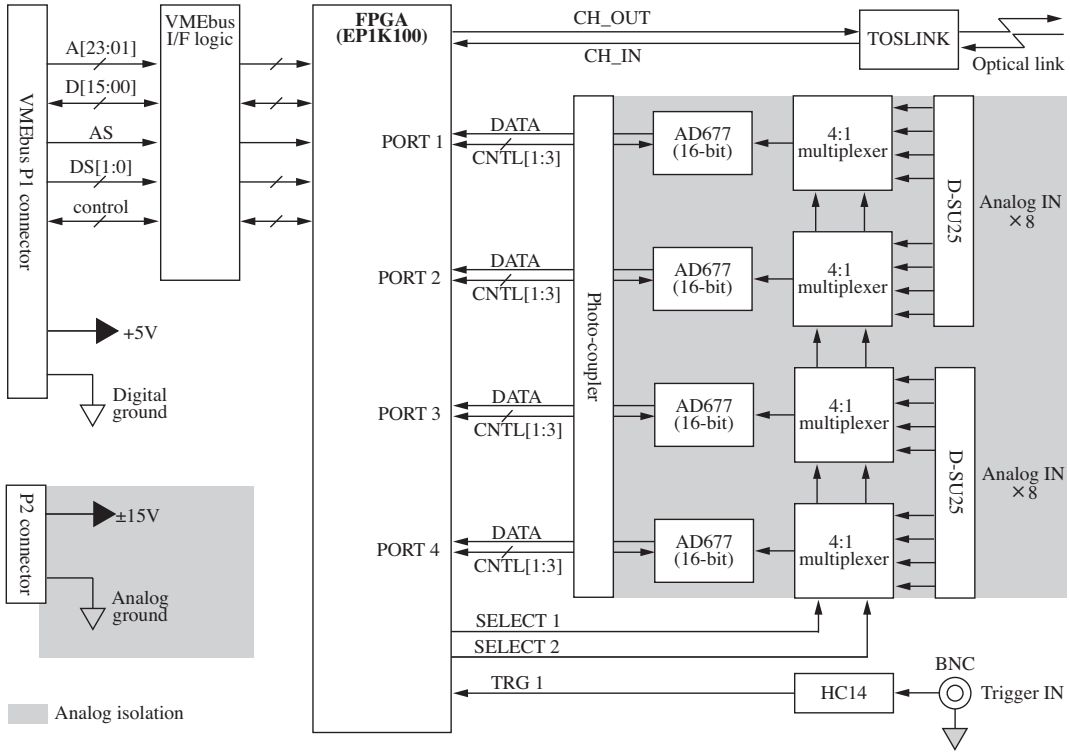


Fig. 3.9. Fig. (a) shows a picture of the HIMV-714 AI-16 board, and Fig. (b) shows a block diagram of the board.

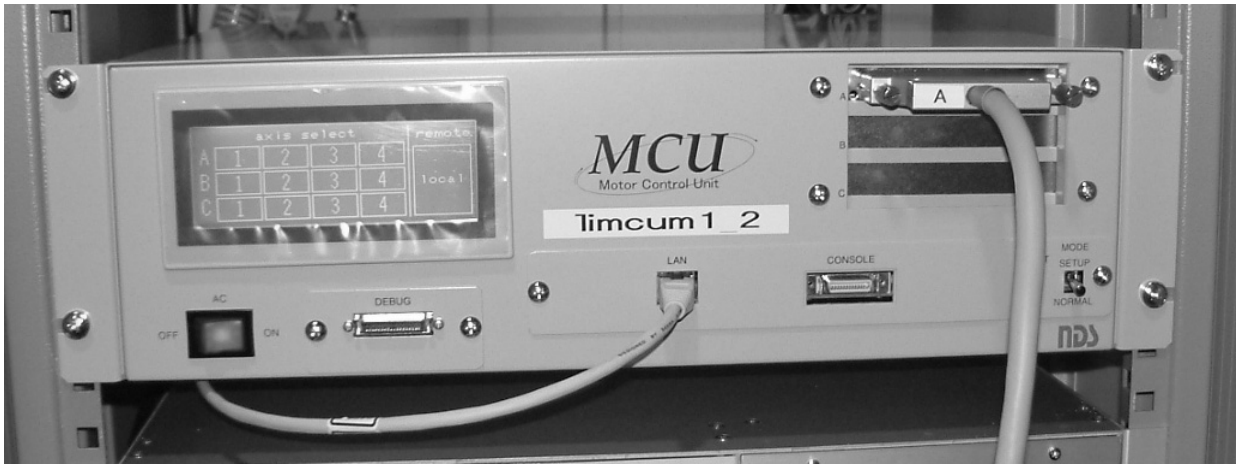


Fig. 3.10. A picture of the MCU (Motor Control Unit).

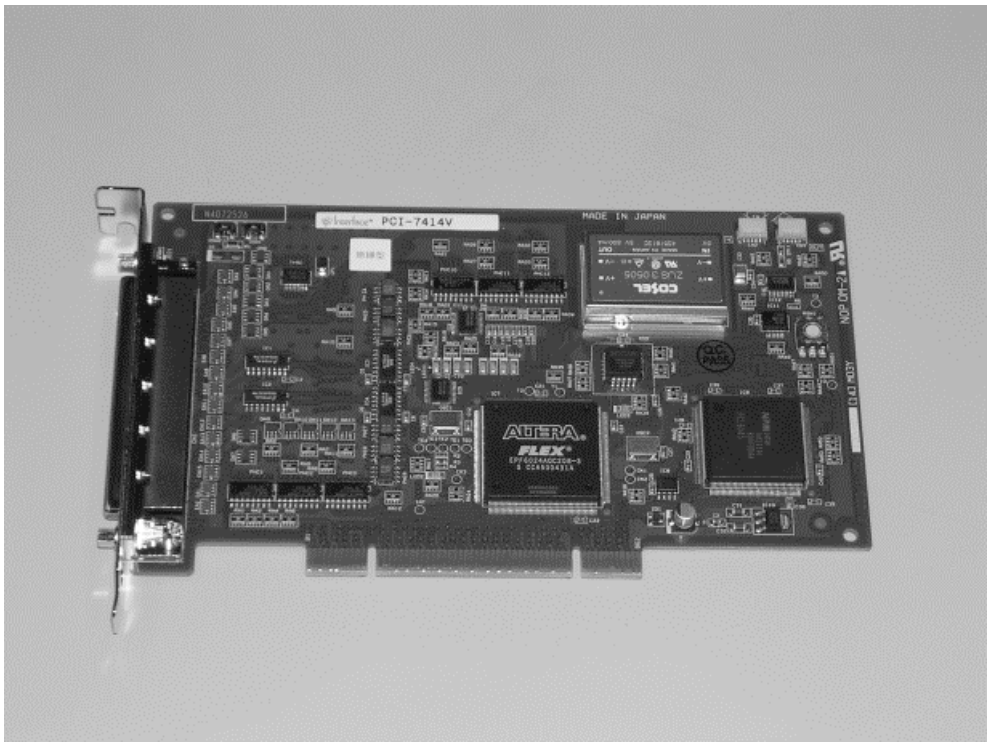
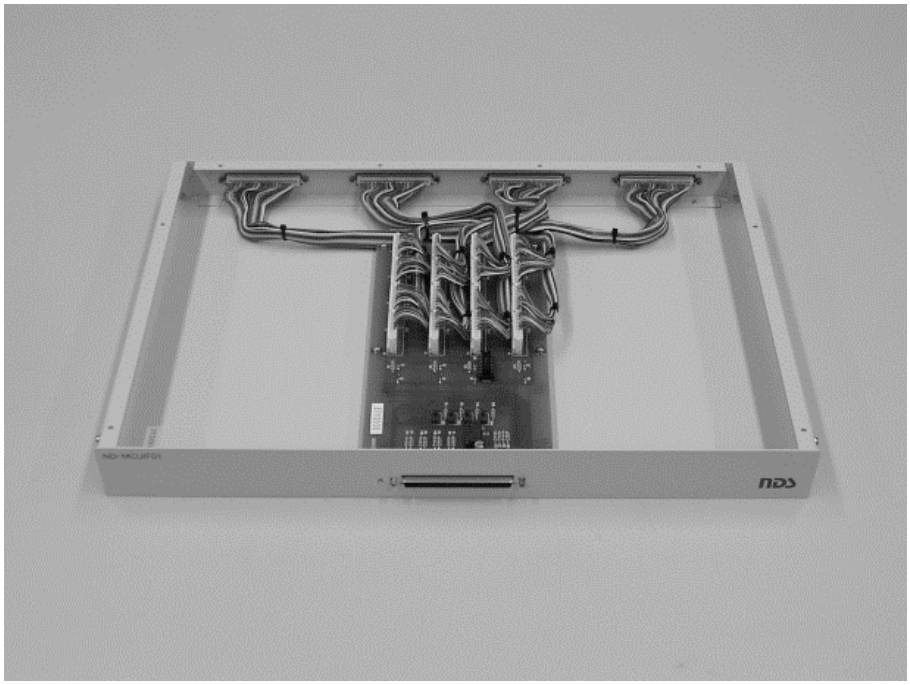


Fig. 3.11. A picture of a PCI-7414V pulse motor control board.

(a)



(b)

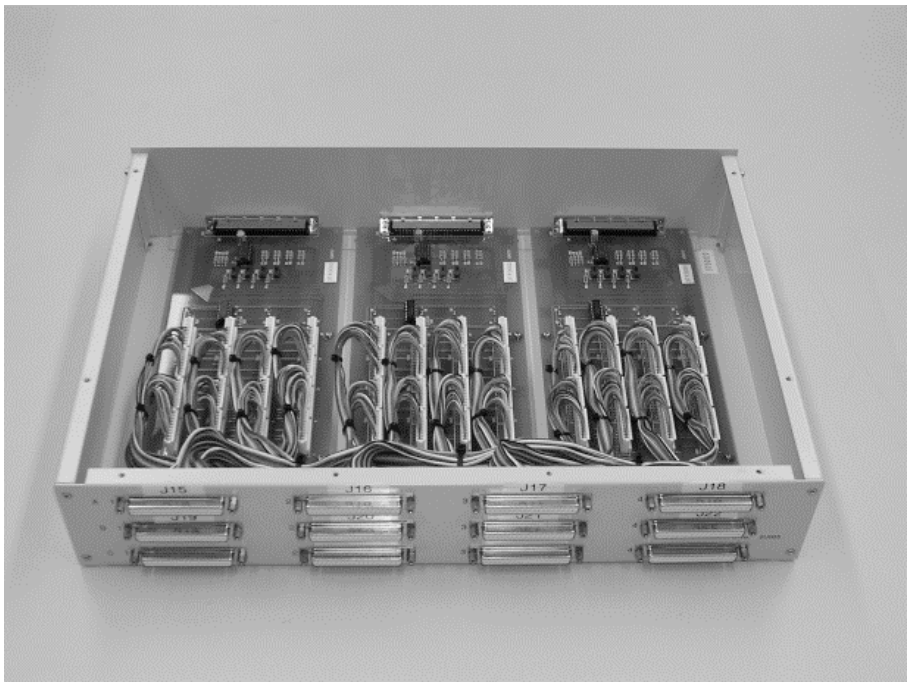
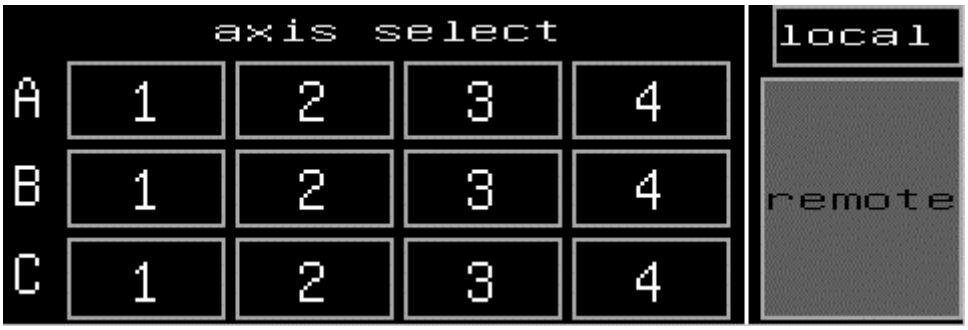


Fig. 3.12. Pictures of newly developed interface box between an MCU controller and an existing signal cables. Fig. (a) shows ND-MCUIF01 which can interface with four axes, and Fig (b) shows ND-MCUIF02 which can interface with 12 axes.

(a)



(b)

Fig. 3.13. Fig. (a) and Fig. (b) show an axis-selection panel and an operation panel on the LCD in the drive mode respectively.

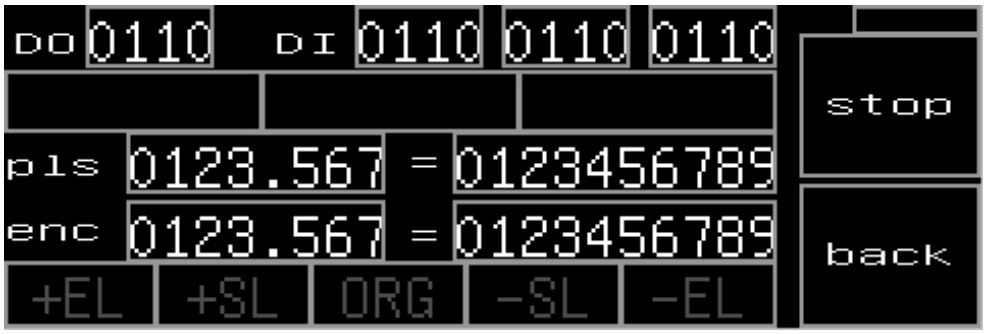


Fig. 3.14. A status view panel of the MCU. This panel displays output pulse counts and encoder counts, limit switches status, result of initialization and so on.

Fig. 3.15. A jog control panel on the MCU LCD touch panel only prepared for a local operation. Fast and slow control buttons are prepared.

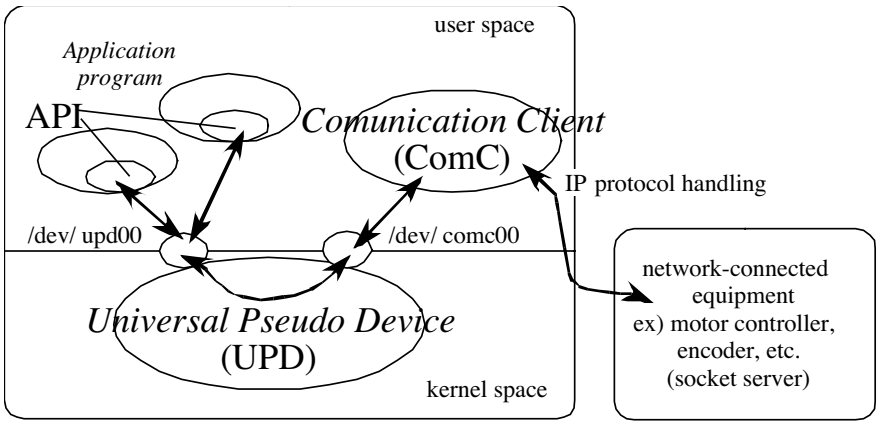


Fig. 3.16. A schematic diagram of the Device Masquerade.

(a) (b)

Fig. 3.17. An exterior of the old connector-box. Fig. (a) is a front view and Fig. (b) is a rear view of the old connector-box.

(a) (b)

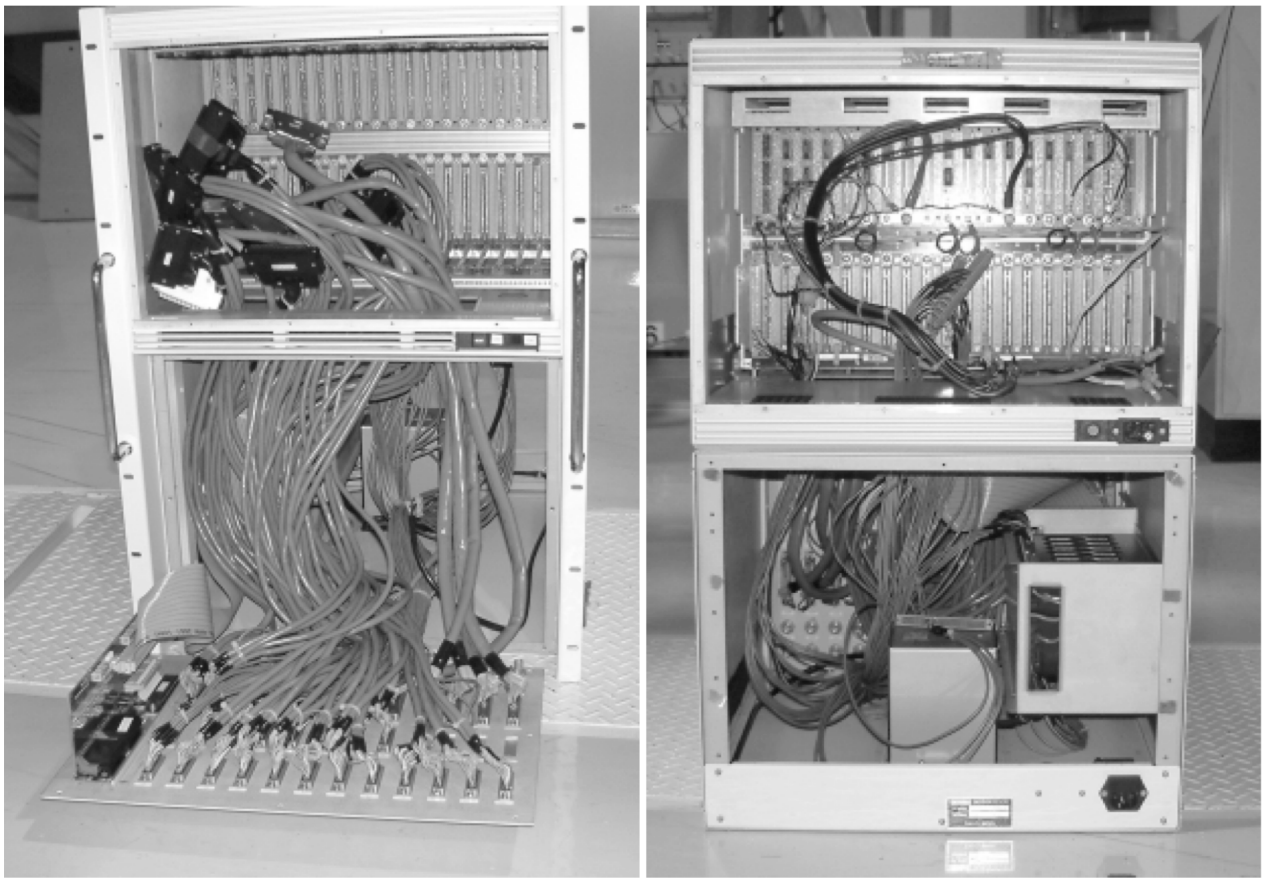


Fig. 3.18. The inside of the old connector-box. Fig. (a) is a front view and Fig. (b) is a rear view of the inside of the old connector box.

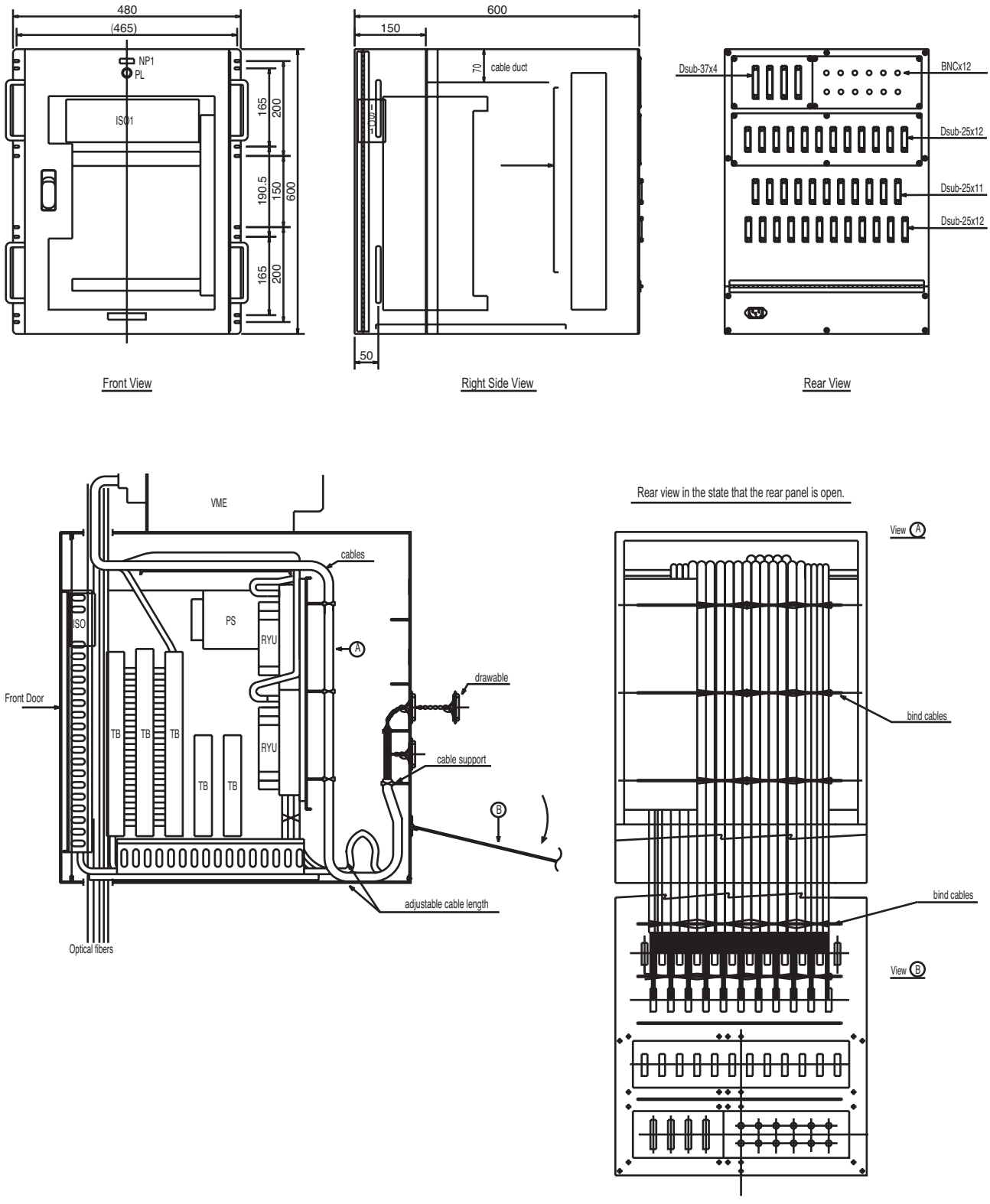


Fig. 3.19. Plans of the new connector-box.

(a) (b)

Fig. 3.20. Appearance of a new connector-box. Fig. (a) is a front view and Fig. (b) is a rear view of the new connector-box. The rear panel is used as a connector panel to attach signal cables from equipment.

Fig. 3.21. The inside and back of the front door of the new connector-box. There are three kinds of terminals in the picture. A normal terminal for analog signals of the magnet power supplies is settled on the side-wall. A terminal of the back of the front door is an isolation-base terminal for analog signals except for magnet power supplies and WGMs. Four terminals on the inner wall are relay terminals for DI and DO signals except for the magnet power supplies.

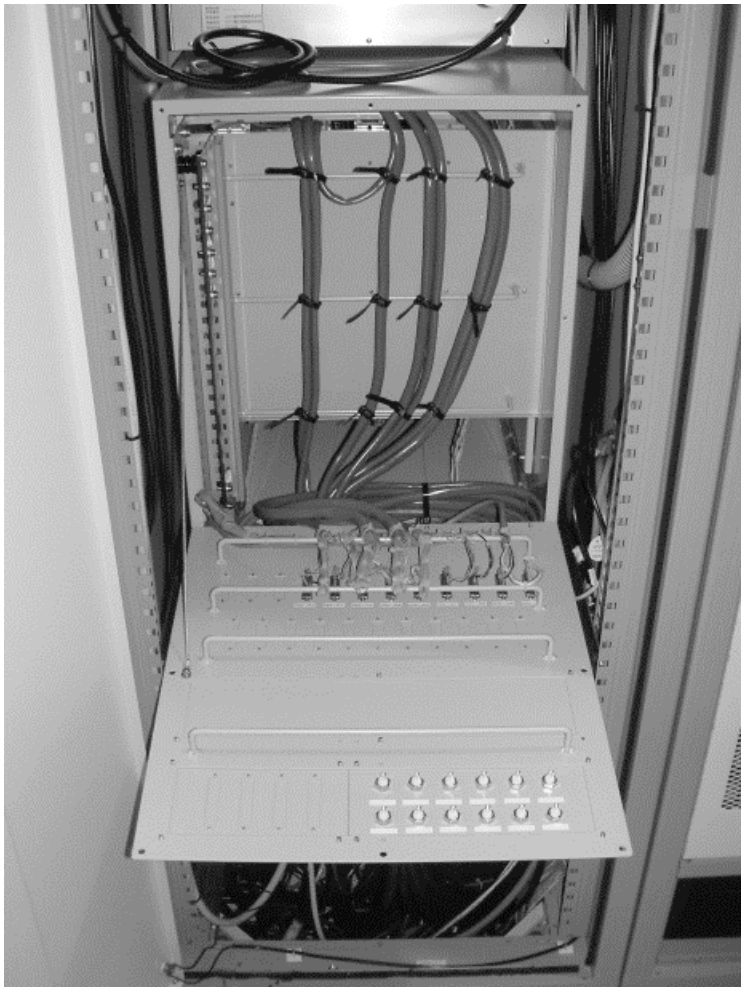


Fig. 3.22. The inside of the rear panel of the new connector box. There are seven bars to tie signal cables.

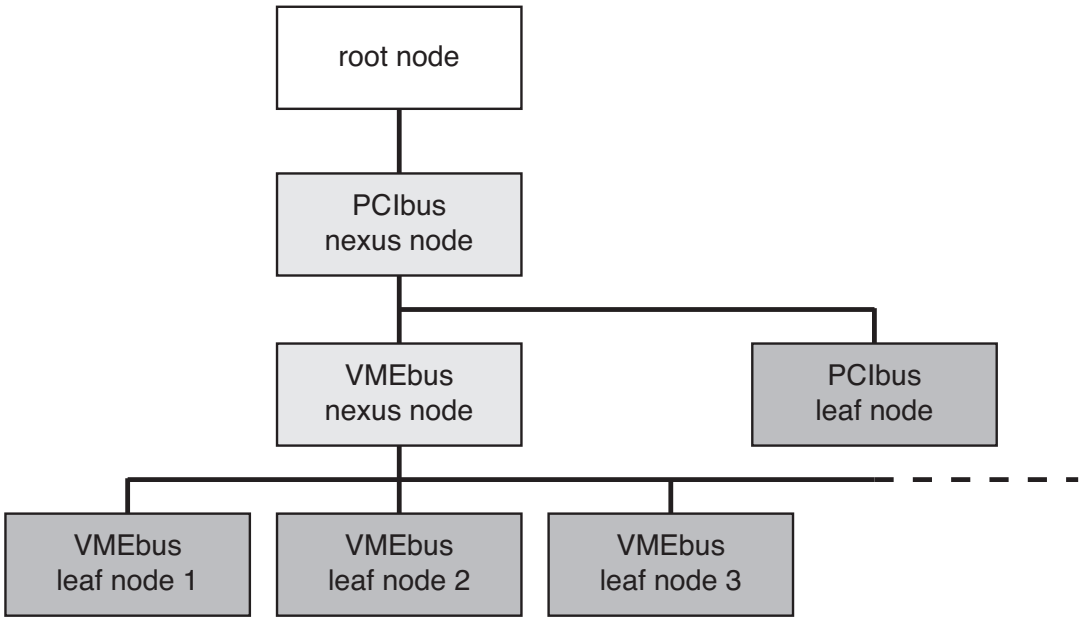


Fig. 3.23. A hierarchy structure of the Solaris device driver.

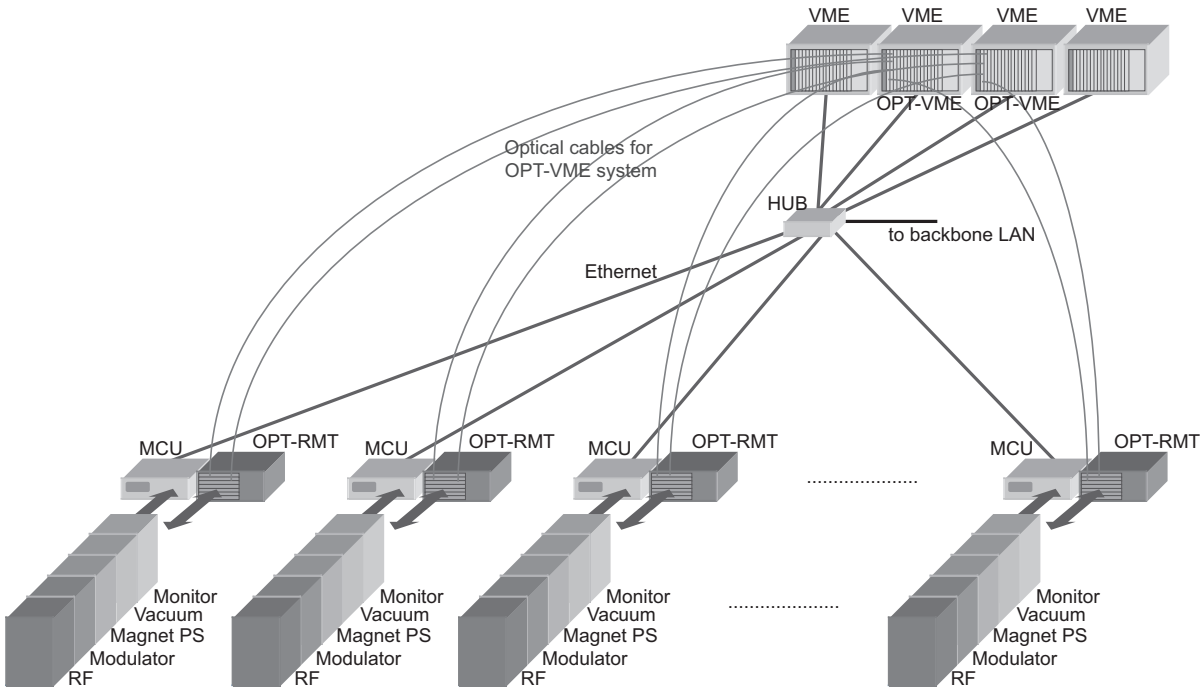
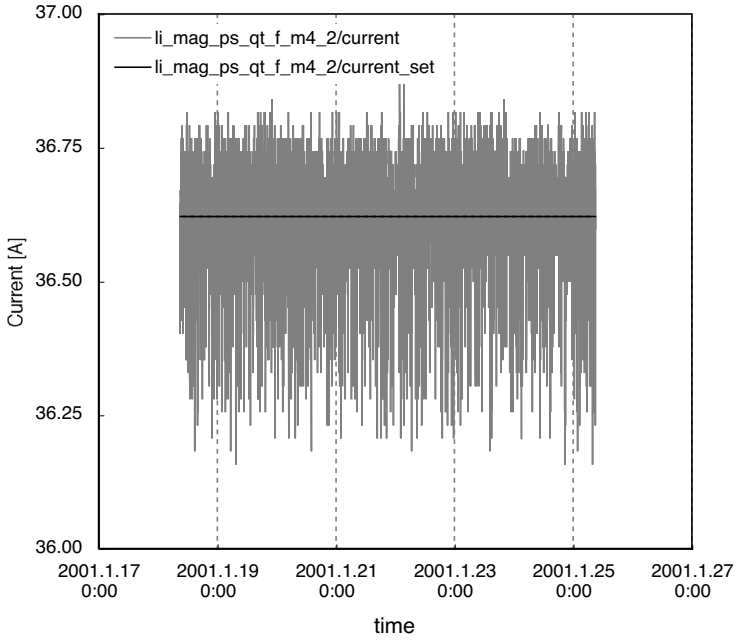


Fig. 3.2. A schematic view of the equipment level of the new linac control system. OPT-RMT boards and MCUs are deployed close to the equipment and VME computers are placed in good environment which is apart from the equipment.

(a)



(b)

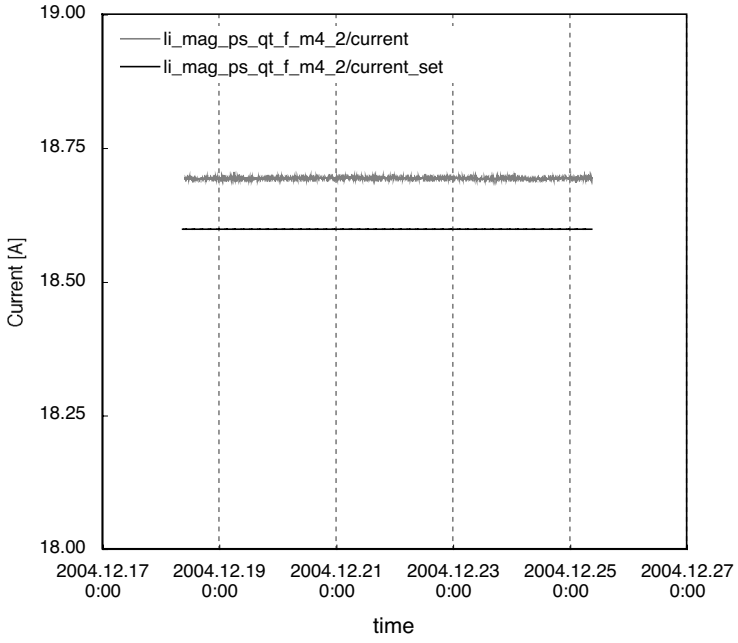


Fig. 3.24. Comparison of noise levels of output current monitor data of a magnet power supply M4-2 QT-F. Fig. (a) shows acquired data with the old linac control hardware. The noise level was about ±2.0% (peak-to-peak) of the set value. Fig. (b) shows acquired data with the new control system. The noise level becomes small, which is less than ±0.11% (peak-to-peak) of the set value.

4. Development of the event-synchronized data acquisition system for SPring-8 linac beam position monitors

4.1. Background of the development

As already described in section 2.1.3, 47 non-destructive BPMs had been installed in the linac and three beam-transport lines. In order to investigate beam dynamics precisely and to stabilize the beam trajectory and energy, a data from BPM system has to be read by synchronizing to the beam shots.

Each BPM detector module has four channels. A channel outputs the 16-bit TTL level data together with a 1-bit inhibition signal. When a trigger signal synchronized to the gun trigger starts the analog-to-digital conversion cycle for the detector module, the inhibition output signal changes its logic level from zero to one. When the data conversion is completed 0.5msec after the trigger input, the inhibition signal waits 0.1msec more and then returns to zero level. Thus the falling edge of the inhibition signal signals that the BPM data are valid and ready for readout. The detector module provides acquisition capability of shot-by-shot beam positions even at a 60pps operation which is the maximum repetition rate of the linac. Currently, the practical repetition rate of the beam operation is limited to 10pps or less in order to lengthen the lifetime of linac components, e.g. thyratrons and klystrons. The nominal injection rate to the storage ring is 1pps.

The following specifications were established for the linac BPM data acquisition system:

1. The system has to capture the shot-by-shot data from 47 BPMs at a practical 10Hz rate.
2. The resulting real time data has to be archived in the standard SPring-8 databases, mainly for the off-line analysis.
3. The system has to be developed as a standard framework to provide application capability to the 60Hz data acquisition in the SCSS linac.

In order to meet these requirements, a new event-synchronized data acquisition system has been developed for the linac BPM based on MADOCA [9], which is the SPring-8 standard control system. The development method, expansion of the existing framework, brings advantages of shortening of the development time, easy debugging and stabilization of the system. This method is effective approach to add a MADOCA-based real-time data acquisition system.

4.2. Hardware of the data acquisition system

Fig. 4.1 shows the hardware configuration of the event-synchronized data acquisition system for the linac BPM [61]. Six VME computers are distributed along the length of the linac in order to acquire the output data from the BPM detector modules. The OPT-VME system is used for data readout. The VME computers

are connected to 100Mbps Ethernet. The VME systems are connected with a shared-memory network (SHM-net) to provide for low-latency data transmission, data sharing and synchronization between software tasks. A PC linked to the same SHM-net collects all of the BPM data on the shared memory boards (SHM-board), and writes the data to the database. A new dedicated database server is introduced into the system so as not to overload the main database server of the SPring-8 control system.

4.2.1. OPT-VME system

Since the outputs of the detector modules of the BPM systems are TTL-level digital signals, the remote I/O system must be located rather close to the sources. Because the output data rate from one BPM detector module could be 1kHz at the most, the remote I/O system has to transfer one BPM data, (16bits + 1bit) × 4 sets, at 10kHz.

The HIMV-615 consists of four sets of the TTL-level digital inputs (16bits + 1 strobe bit). Fig. 4.2 shows the HIMV-615 remote board. The four connectors for the TTL-level digital inputs can be seen on the front panel. One board is used to read one BPM data. The HIMV-615 communicates with the HIMV-658A OPT-VME master board using the read-only mode. The HIMV-615 sends the signal amplitude from four BPMs to the master board in less than 20μsec.

4.2.2. Shared memory network

An SHM-net enables to share the hardware memory images between SHM-boards connected to the same network. The SHM-net system uses six Advme1522A boards for the VMEbus computers and one Adpci1523A board for the PCIbus computer. The SHM-net system [62] uses Fibre Channel layers FC-0 and FC-1; the transmission speed is 250Mbit/sec. Since the memory mirroring is done transparently by the SHM-board hardware, the overlaying software does not need to take any particular details of the communications into account. The effective speed of the data transfer is 4Mbyte/sec. A multi-mode optical fiber with 50μm core and 125μm clad is used for the link and the distance between the nodes can be up to 1km. A single board can provide up to 8Mbytes of memory at the maximum and one SHM-net can support up to 255 boards.

An SHM-board can issue an interrupt command to a specific node or to all nodes on the same network. The board that receives the command generates an interrupt to the host CPU through the computer bus. Further an arbitrary 4-bit data nibble to be passed as additional information with the interrupt command can be attached. The interrupt can be used to synchronize the control software processes running on each CPU.

Fig. 4.3 is a block diagram of the SHM-board. There are three types of FIFOs (first-in first-out) buffering the data frames on the board. They are T-FIFO for data transmission, R-FIFO for data reception and I-FIFO for generating an interrupt to the computer bus. Each FIFO has a depth of 512 data frames. The SHM-board

is designed to guarantee that the data-frame transmission latency in one node is less than 1.4μsec if all the FIFOs are empty. The scheduled data frame in the T-FIFO is transmitted to the next node within 960nsec. The scheduled data frame in R-FIFO is processed in 200nsec. The delay through the optical fiber cable is estimated to be about 4μsec per 1km [63]. By taking these latencies into account, individual memory regions to each VME system can be assigned in such a way as to avoid any data update conflict.

4.2.3. VME computers

The six distributed VMEbus systems are equipped with one Advme1522A SHM-board and several HIMV-585A boards for data acquisition from the individual BPM detector modules. For the CPU, Advm8001 [62] boards are used. The specifications of the Advme8001 are listed in Table 4.1. The Advme8001 is one of the IA32 architecture CPU boards already deployed in the SPring-8 control system. For the operating system, the Intel-based Solaris 7 is used. Table 4.2 shows hostnames of the VMEbus computers and the numbers of associated BPMs.

4.2.4. PC and database server

At the top of the linac BPM data acquisition system, two PCs are used. One PC is the overall controller and prepares the accumulated data for storage in the database. The other PC is a general-purpose relational database server, here a Linux-based Sybase system. This server is dedicated to the BPM system.

The controller PC is equipped with an Adpci1523A SHM-board, which links it in with the SHM-net shared with the VME computers. The PC currently in use has an 850MHz Pentium III processor with 256kB L2 cache, and 128MB of DRAM. RedHat Linux 6.2 for the OS was chosen because gateway interface software to the database and a device driver for the Adpci1523A were both readily available.

The database server is now running on a 2.0GHz Intel Xeon processor computer. This server conforms to the SPring-8 control database schema and is linked to the main database server by using distributed database technology. The database system is described in the section 2.2.2.7 and section 4.3.3.

4.3. Software development for the event-synchronized data acquisition system

4.3.1. Development of the event-driven data acquisition software framework

In order to realize the event-synchronized data acquisition system for the linac BPM, it was necessary to develop the required event-driven data acquisition software by extending the existing SPring-8 control software framework MADOCA.

The MADOCA framework, based on the client-server architecture as described in the section 2.2, does not support interrupts (call-backs) from the server software to the client processes. Therefore, data collection software could not start by getting an event at the hardware level. By expanding functionality with the EMA, a new event-driven software framework has been developed. The new software consists of event-driven EMAs (EMA-EVs), event generators (EVGENs) and data fillers (Fillers), as illustrated in Fig. 4.4 [61]. A set of the software processes that performs data acquisition synchronized to a common event forms one *project*. A *project* consists of some EMA-EVs and one Filler process. The composition of a *project* is described in a *project file* as shown in Fig. 4.5. The *project file* has a tag format like XML (Extensible Mark-up Language). One set of <project> and </project> tags represents the *project*. Also a corresponding <filler></filler> tag is defined for the Filler, <emaev></emaev> tags for the EMA-EVs and <signal></signal> tags for acquired signals in the same fashion. With each tag, there are some elements that define properties such as the size of an associated ring buffer, signal ID number, data type, and executable filename, etc.

The Filler process is the lead process which works as the project manager. When the Filler starts, it reads the *project file* and creates a data table for the *project* in the SHM-net memory. The data table contains a data record area for each EMA-EV, these are ring-buffers, and the size of the buffers is assigned in the *project file*. Fig. 4.6 shows an example data table structure. By locating the data table in the SHM-net memory space, the processes running on computers that belong to the same SHM-net can share information via the table. After the Filler starts, it controls the EMA-EVs in the same *project* by sending messages such as *start*, *stop*, *create,* and *destroy*. And the Filler cyclically stores the shared memory table data sets into the persistent relational database along with a time stamp and an event number.

The EM creates the EMA-EV in the same manner as the EMA, and the EM controls it by sending S/V/O/C command messages. After the EMA-EV starts, it reads its *project file* and stores the list of the signals for which data should be taken in memory. The listed signals are described in the S/V/O/C command format. After reading the *project file*, the EMA-EV creates a process called EVGEN. When the EMA-EV gets a *start* message, it directs the EVGEN to start watching the hardware event and waits for event messages from the EVGEN. When EVGEN detects the specified event, it sends the event message to the EMA-EV together

with the event number. As soon as the EMA-EV receives the event message, it starts to acquire all the listed signals. The collected data are written into the data table by using the event number as a key. At the same time, the acquired time is also included in table data. After writing a set of data, the EMA-EV updates the newest record number referred to by the Filler.

When the EMA-EV receives the *stop* message, it instructs the EVGEN to stop the event watching. In this state, the EMA-EV can restart the data acquisition by accepting a *start* message again. The data acquisition can be completed by sending the *destroy* message to the EM, and the EMA-EV terminates after receiving this message from the EM.

The EMA-EV controls the start and stop of the EVGEN by using the UNIX signal mechanism. This is because the direction of the S/V/O/C message is limited to be unidirectional, from the EVGEN to the EMA-EV.

Since the method of watching an event depends on the event source, EVGEN provides an abstraction for event detection as a user-defined function.

4.3.2. Process synchronization with the shared memory network

In order to synchronize all the EVGENs, the interrupt function of the SHM-net is used. One master EVGEN in the data acquisition system uniquely monitors a lead hardware event. When the master EVGEN detects the hardware event, it generates an interrupt to all the SHM-boards on the SHM-net. Other slave EVGENs may wait for the interrupt from the SHM-board as events, so that they are awaken by the interrupt at the same time. Then, all the EVGENs can send event messages to the EMA-EV on each computer respectively. Thus all the EMA-EVs can be synchronized with a hardware event.

For event identification, each EVGEN maintains the event number as an internal parameter. The EVGEN sets the event number to zero at creation time, and increments the event number whenever the event occurs. In order to preserve consistency between the event numbers of all the EVGENs, the additional information passed with the SHM-net interrupt command is utilized. When the master EVGEN sends the interrupt command, it attaches the lower four-bits of the event number. A slave EVGEN can correct its event number by comparing its internal event number to the interrupt parameter. The EMA-EV stores its collected data in the ring buffer created on the SHM-net, the address is calculated from the event number. If there is a lost event, the EMA-EV writes a fail data flag (see Table 2.4) to the event area.

4.3.3. Development of the data logging software

The Filler continuously watches the newest event record numbers on the SHM-net and writes all the updated data into the database of the SPring-8 control system. After storing the data sets into the database, the Filler updates the read record number on the SHM-net.

Fast, frequent data acquisitions cause heavy loads on a database system. Putting heavy loads on the main database server of the SPring-8 control is not allowed because it could interfere with timely data access needed for proper accelerator operation. To spread the load while still satisfying common data access requirements, a distributed database system, which is also described in section 2.2.2.7, was introduced to support fast data acquisition [41]. Building a dedicated remote server enables to perform the fast data insertion while producing only minimum load on the main server.

To achieve faster data insertion, a single text field is used instead of the real number fields. That is, the array of real numbers is simply preprocessed by forming a collected data vector into a text string. Up to 2GB of text can be stored in one field of a Sybase ASE database table. Table 4.3 shows the performance improvement in data insertion by using this method. The stored raw data can be retrieved and reformatted for display by web browsers, or by C-language function calls. A typical web browser page can display graphs of all the BPM data from each shot and signal time-transition from any one BPM.

4.4. Performance measurements

Beginning in December 2003, the event-synchronized beam data acquisition from the linac 47 BPMs started. Since then the system has been working well. All the BPM data collection from each beam shot and storing them in the database worked satisfactory. The system performance has been measured at the normal linac 1pps operation and also at 10pps.

4.4.1 Setup for the measurements

An EVGEN running on VME computer libpmh0 was assigned to be the master. In order to avoid any events being lost by the Filler process, plenty of record space was allocated on the SHM-net, i.e. the buffer length was defined to be six hundred.

Solaris is a real-time OS, which has a pre-emptive and multi-thread kernel as already described in the section 3.3.2. The Solaris scheduler determines the scheduling of software process based on global priority levels (see Fig. 3.4). Deterministic operation, and real-time execution, of the polling software processes is necessary for proper event detection and processing. Therefore, the global priority level of 100 was assigned to the data acquisition software, EMA-EV, EVGEN, EM, and MS, in the RT class of Solaris 7. When a process of higher global priority becomes ready to execute, the current running process is pre-empted and the higher process is executed immediately. Commonly, UNIX software processes run in the TS class, which is the traditional process schedule mechanism in UNIX. In the TS class, the system scheduler dynamically changes the process priority in order to maximize system throughput. Consequently, the execution of software process may fail to be deterministic. On the other hand, software processes running in the RT class

are scheduled with fixed higher priority than the system tasks. Therefore, no processes except interrupts can pre-empt the data acquisition software.

For each BPM, eight data points are taken, which are four output voltages of the detector module, horizontal beam position X, vertical beam position Y, and error and average of the voltages. These values are simply expressed as following equations:

$$X = C_x(V1 - V2 - V3 + V4), \quad \text{..... (4.1)}$$

$$Y = C_x(V1 + V2 - V3 - V4), \quad \text{..... (4.2)}$$

$$error = \sqrt{C_x C_y}(V1 - V2 + V3 - V4), \quad \text{..... (4.3)}$$

$$average = \frac{(V1 + V2 + V3 + V4)}{4}, \quad \text{..... (4.4)}$$

where,

V1, V2, V3, V4: output voltages of the detector modules.

Actually, the output voltages from the detector module have to be corrected using attenuation factors of feed-through.

4.4.2. Measurements and results

Measurement at 1pps operation

Fig. 4.7 shows the horizontal and vertical trajectory of the electron beam for one pulse captured during 1pps operation, and Fig. 4.8 exhibits the time variation in horizontal beam position at one BPM. These data were acquired at accelerator tuning time.

There were 8044 events of the electron gun fire at 1Hz interval during the measurement, and it was successful in capturing 99.9% of all the events as shown in Figs. 4.9. The horizontal axes of these histograms mean time intervals between consecutive captured events. If all the events can be captured without any drops, all the intervals should show a 1sec separation during the 1pps operation. If any events have been dropped, the time intervals will become equal or greater than 2sec. Fig. 4.9(a) shows the results from one BPM collected by host libpmm20 which has a large number of signals to acquire. Fig. 4.9(b) shows the collection result of one BPM from libpmh0 which deploys the master EVGEN. Since Figs. 4.9(a) and (b) have exactly the same shape – the same number of lost events, it is concluded that the event loss occurred at the same time, independently of individual VME system loading.

Measurement at 10pps operation

Currently the SPring-8 linac operates with beam acceleration cycles up to 10pps. Accordingly, it has been confirmed that the event-synchronized data acquisition system could feasibly collect the full BPM data when in 10pps operation.

Fig. 4.10(a) shows the time interval of one BPM data acquisition collected by the libpmm20. About 99.9% of the 4683 events were successfully collected at the 10Hz. This result indicates that the 600MHz Pentium III has enough CPU processing power to acquire 96 signals within 100msec. Fig. 4.10(b) exhibits the acquisition result of one BPM by the libpmh0. Again, the event losses show the same characteristic, as shown in Figs. 4.10

4.4.3 Discussions

Measurement results in the previous section imply that the master EVGEN caused all of the event losses. It is suspected that the master EVGEN failed to detect some events because of task switching done by the Solaris process scheduler. One possibility might be that interrupts generated by hardware such as a network device caused an untimely pre-emption. Since hardware interrupts are processed with the highest priority under Solaris, even the master EVGEN running in the RT class can be pre-empted. If the pre-emption timing coincides with the inhibition signal, EVGEN might miss the event detection. It is thought that the 0.6msec width of the inhibition signal is too short to be detected every time for all events when there might be delayed sampling caused by task switching.

In order to avoid event detection losses, the master EVGEN should detect the hardware event by interrupt instead of the polling.

4.5. Studies of the performance improvements

4.5.1 Approaches to the performance improvements

The measurement results in the previous section showed that event detection efficiency of the BPM data acquisition system had to be improved. As discussed above, the master EVGEN should detect the hardware event by an interrupt to the VME CPU board instead of the polling. The interrupt enhances detection efficiency of the hardware event because the interrupt has highest global priorities in the Solaris OS. Therefore, a VME interrupt board was newly installed in the libpmh0, on which the master EVGEN ran, to generate an interrupt to the VME CPU board by receiving gun trigger signal. The master EVGEN was modified to wait for an interrupt by the interrupt board instead of the polling.

From the measurement results in the section 4.4, the maximum repetition rate of 96 signals acquisition by the host libpmm20 was estimated to be approximately 30Hz. The consumed time was dominated by the software framework processing time, that is, CPU execution time. The following improvements were available to achieve the 60Hz data acquisition, which was the maximum repetition rate of the linac beam operation.

1. Replace the current CPU boards by two times faster CPU boards.
2. Add one more CPU board to form dual-master CPU configuration.
3. Optimize the software framework to realize faster processing.

The first and the second approaches were suitable to fit the present software framework by the least modification. The third approach would require much time for improvement and debugging of the software framework. Considering the future use in the SCSS linac, the first approach was preferable to realize more powerful processing in the data acquisition system. As already described in section 3.4.1, the success in developing the Universe II nexus device driver provided capability of choice of the first approach.

As a new CPU board, SVA041-185T [65] with 1.8GHz Pentium M was adopted. The new CPU board was 2.8 times faster than that of Advme8001. The specifications of SVA041-185T are listed in Table 4.4. For the operating system, the Intel-based Solaris 9 was newly introduced.

4.5.2 Measurements and results

The performances of the modified system were measured under the same condition described in the section 4.4.1 except the event detection method, the VME CPU boards and the version of the OS.

Measurement at 1pps operation

Figs. 4.11 show the measurement results of the modified system under the 1pps linac operation. Fig. 4.11(a) (Fig.4.11(b)) shows the time interval of one BPM data acquisition taken by libpmm20 (libpmh0) respectively. The improved system succeeded in capturing all the BPM data at all the 13273 events of electron gun fire at 1Hz interval. These measurement results indicate that the hardware interrupt event detection method brought the complete event detection.

Measurement at 10pps operation

Figs. 4.12 represent the results of the BPM data acquisition under the 10pps operations of the linac. A total of 6371 events were recorded, and the modified system succeeded in capturing all the events without any losses, as shown in Figs. 4.12.

Measurement at 60pps operation

At first, the insertion speed of all the linac BPM data into the database by the filler process was measured. The filler process was expected to insert a set of synchronized data into the database with more than 60Hz speed because the filler process employed a single text insertion method as described in the section 4.3. The filler process took about 50msec to insert the data. A slow speed indicated that the database server had the data insertion latency. Therefore, it was found that 20Hz was the maximum repetition rate in the consecutive data acquisition of the present system. If the system consecutively takes the BPM data at faster event rate than 20Hz, i.e. 60Hz, the data insertion by the filler process can't catch up with the event, and the first recorded data on the SHM-net are overwritten before the filler inserts all the data.

In this measurement, the size of the ring buffers on the SHM-net for a VME system was set to 600. This means the data taken at the 60pps operation was kept on the SHM-net for 10 sec. The performance of the modified data acquisition system was measured under the intermittent 60pps operation of the linac. In actual measurement, the linac operated 5 sec at 60pps and paused 25 sec, then repeated the same operation cycle. The filler process took 15 sec to insert all the collected 300 sets of data on the ring buffer during 5 sec operation.

Figs. 4.13 exhibit the measurement results of the improved data acquisition system at the intermittent 60pps operation. The improved system succeeded in capturing all the BPM data at all the 11370 events of the electron gun fire at 60Hz.

Presently, the latency of the insertion into the database is a bottleneck for the consecutive 60Hz acquisition. The faster insertion technique such as a bulk insertion is available, as shown in table 4.3. It is a promising technology to achieve the consecutive 60pps data acquisition of the linac BPM.

4.6. Applications for the beam stabilization in the linac

The linac BPM data acquisition system has been applied to an actual linac operation since December 2003. It has continuously acquired all the BPM data for every shot and has accumulated all the acquired data into database throughout the linac operation including injection both to the booster and the NewSUBARU storage ring.

The accumulated BPM data is available for data analysis by retrieving arbitrary BPM data from the database. Fig. 4.14(a) exhibits horizontal beam trajectories along the length of the linac at 10pps operations, and Fig. 4.14(b) shows their variations. The trajectory variation was caused by a kicking perturbation at the injector section where beam energy was about 60MeV. The analysis is available to retrieve a time-specified beam shot data using a web browser.

In addition to off-line analysis, the BPM data can be applied to on-line automatic feedback control for beam trajectory and beam energy stabilization.

4.6.1. Automatic beam position feedback control

A beam trajectory drift in the linac makes injection efficiencies to the booster and the NewSUBARU storage ring worse. A room temperature variation causes the trajectory drift. Since the SPring-8 top-up operation highly requires constant injection current, the linac beam trajectories have to be as stable as possible. In order to achieve such stable injections, a beam position feedback control program was introduced in September 2004 [64]. Before introducing the feedback control program, operators manually corrected the beam trajectory in every two or three days. This program stabilized beam positions and angles at an injector section located upstream of the linac, a matching section located downstream of the ECS, and a beam transport (BT) section to the NewSUBARU. These sections have two sets of steering magnets for x and y-plane in the drift space, which is located upstream of two non-dispersive BPMs. Fig. 4.15 represents the matching section and the BT section to the NewSUBARU. Beam positions and angles to the booster are finally stabilized at the matching section, and those to the NewSUBARU are finally stabilized at the BT section.

The position feedback program cyclically retrieved the BPM data from the database, calculated correction values for the steering magnets to maintain the beam positions within a tolerance, and applied the correction values to the magnets. The beam position tolerance is currently set to ±30μm and the feedback control cycle is set to 5 times of the injection interval. In the case of an every minute injection for several bunch operations at the SPring-8 storage ring, the feedback control cycle was set to 5 minutes.

Fig. 4.16(a) shows time variation of horizontal beam position at a LSBT-1 BPM in the matching section (see Fig. 4.15) before introducing the automatic beam position feedback control. Fig. 4.16(b) shows a result of horizontal beam position drift at the same BPM after applying the feedback control. The comparison of the Fig. 4.16 indicates that the position feedback control program succeeds in stabilization of the beam position in the matching section.

4.6.2. Automatic beam energy feedback control

As describing in 2.1.2, the SPring-8 linac has the ECS which is located upstream of the matching section (see Fig. 4.15) in order to compensate accidental energy variation and to compress energy spread of injection beam to the booster and the NewSUBARU. However, unexpected small and slow energy drift at the downstream of the ECS was observed. It was caused by a room temperature variation. The energy drift affected the beam injection efficiency both of the booster and the NewSUBARU. In order to stabilize the beam energy, a beam energy feedback control program was introduced as a supplemental system of the ECS in September 2004.

The energy feedback program retrieves the accumulated BPM data from the database and adjusts the ECS phase shifter as the horizontal beam position within a tolerance at the LSBT-3 BPM, which is installed in a

dispersive section located downstream of the ECS (see Fig. 4.15). The adjustment was executed only when the beam positions in the matching section exceeded the double tolerance defined by the position feedback control. The tolerances of the position feedback control and the energy feedback control are currently set to ±30μm and ±300μm (~ ±0.03% of beam energy), respectively.

Fig. 4.17(a) shows time variation of horizontal beam positions at a LSBT-3 BPM before introducing the automatic beam position feedback control. Fig. 4.17(b) shows a result of horizontal beam position drift at the same BPM after introducing the feedback control. Fig. 4.16 indicates the energy feedback control program succeeded in stabilizing the beam energy.

4.6.3. Investigation of unexpected burst current emissions

On 5 October 2005, a phenomenon that the linac electron gun very occasionally emitted unexpected burst current was found. Fig. 4.18 shows the burst current observed by using the beam current monitors installed just behind the buncher section and the electron gun. The burst current was emitted 40nsec after the normal 1nsec beam and it had the length of 2msec. The total charge of the burst current was nearly 10nC. Beam current monitors for radiation safety control to limit the transported charge to the NewSUBARU storage ring couldn't record the burst current because these monitors measured the normal 1nsec beam with a narrow gate signal. Therefore, investigation of the burst current had to be made how many times the phenomena occurred in the past and whether the burst currents were injected to the NewSUBARU storage ring.

The BPM data collected by the shot-by-shot data acquisition system were utilized for the investigation because the signal processor of the linac BPM system didn't use the gate signal to the 1nsec beam and the shot-by-shot data acquisition system accumulated all the acquired BPM data into the database.

Table 4.5 shows the result of the investigation since November 2004. It was found that there occurred 35 times of the burst currents. The charge of the burst current at the downstream of the ECS was 5.8nC ~ 8.5nC before a cathode exchange in the summer of 2005 and 8.9nC ~ 9.7nC after the September 2005.

Table 4.5 represents that the burst current toward the NewSUBARU storage ring occurred only once on 13th of November 2004. And the charge of the beam shot was evaluated by using the BPM data, as shown in table 4.6. A beam slit installed behind a bending magnet in the upstream of the L3BT reduced the burst current to about a fifth. And a beam slit installed in the L4BT finally reduced the burst current to 0.5nC or less. This investigation result proved that the injection of the burst current into NewSUBARU didn't become an issue of radiation safety control. Thus the shot-by-shot data acquisition system for the linac BPM played an essential role to trace the entire burst current in the past because the system accumulated all the BPM data synchronizing to all the beam shot.

At present, the burst current emission is not observed as the result of adjustment of the grid voltage of the electron gun. The cause of the burst current doesn't become clear but fault of the grid of the electron gun is suspected.

4.7. Future applications

As described in the above, the linac BPM data acquisition system has been developed as the expansion framework of MADOCA. Therefore, the newly developed event-driven data acquisition framework is easily applicable to other MADOCA-based control systems.

The new framework will be applied to 60Hz data acquisition of the SCSS. In order to overlap the electron beams on the X-ray beams to achieve lasing, trajectories of the electron beams in undulators have to be kept in a straight line. The tolerance will be typically 4 μm transverse for each 4.5m long undulator segment [66]. Correction using corrector magnets based on shot-by-shot beam position monitoring will be necessary to achieve the accuracy. The application to the SCSS data acquisition will help the beam stabilization of the SCSS a lot.

The new framework will be also applied to fast data collection for the new BPM system installed into the SPring-8 storage ring. The new BPM system will provide measurement capability of submicron resolution and a few milliseconds throughput.

Table 4.1 Specifications of the Advme8001 CPU board.

CPU	Intel Low-power Pentium III / 600MHz
System bus frequency	100MHz
Chipset	Intel 440BX
L2 cache	256kB
Memory	128MB
Ethernet	10Base-T/100Base-TX(Intel 82559) × 1ch
PCI-VME bus bridge	Tundra Universe IIB

Table 4.2 VME computers by hostname, showing the numbers of signals acquired by each.

Hostname	CPU board	OS	Number of BPMs	Number of signals
libpmh0	Advme8001-P3	Solaris7	4	32
libpmh5	Advme8001-P3	Solaris7	10	80
libpmm6	Advme8001-P3	Solaris7	6	48
libpmm20	Advme8001-P3	Solaris7	12	96
libpml3bt	Advme8001-P3	Solaris7	8	64
libpml4bt	Advme8001-P3	Solaris7	7	56

Table 4.3 Results of database data insertion timing test. For this test, 400 real numbers were inserted into the table in two ways: as an array of 400 real numbers and after concatenating into one long text string. And this table shows the insertion result of one long text string using bulk copy option instead of sql command. Test done on a 2.0GHz Intel Xeon based server. Results measured as maximum repetition rates.

Method	Insertion speed (Hz)
Insert real array	17.5
Insert text	62.5
Bulk insert text	276.5

Table 4.4 Specifications of the SVA041-185T CPU board.

CPU	Intel Pentium M 745 / 1.8GHz
System bus frequency	400MHz
Chipset	Intel 855GME + ICH4
L2 cache	2MB
Memory	512MB
Ethernet	10Base-T/100Base-TX(Intel 82551ER) × 2ch
PCI-VME bus bridge	Tundra Universe IID

Table 4.5 Investigation results of the unexpected burst current since November 2004. These events were picked out first from an interlock event of the gun modulator by CCG alarm. Then BPM voltages were investigated around the time.

date of occurrence of interlock	time of occurrence of interlock	time of the burst current observed	charge [nC] at LSBT-4 BPM or L4BT-9 BPM	Destination of the beam
2004.11.13	2:51:58	2:51:52	0.45	NewSUBARU
2004.12.23	8:51:07	8:51:00	8.54	Booster
2004.12.25	21:39:20	21:39:14	7.15	Sy
2005.3.18	12:04:03	The electric gun is not triggered.		L2 beam dump
2005.5.11	10:20:05	The electric gun is not triggered.		L2 beam dump
2005.6.26	17:17:41	17:17:37	5.79	Booster
2005.7.11	13:14:08	database trouble		Booster
2005.7.30	3:41:13	3:41:03	7.05	Booster
2005.7.30	4.19.05	4.19.01	6.81	Booster
2005.9.15	13:06:10	The electric gun is not triggered.		L2 beam dump
2005.9.23	4:00:04	4:00:00	8.86	Booster
2005.9.23	15:15:06	15:15:00	9.01	Booster
2005.9.24	0:35:05	0:35:00	8.77	Booster
2005.9.25	9:55:09	9:55:02	9.33	Booster
2005.9.25	16:35:10	16:35:01	9.20	Booster
2005.9.28	19:30:08	19:30:01	9.68	Booster
2005.9.30	19:31:11	19:31:01	9.73	Booster
2005.10.3	2:17:07	2:17:01	9.50	Booster
2005.10.3	2:38:47	2:38:41	9.27	Booster
2005.10.3	3:10:28	3:10:20	9.43	Booster

2005.10.3	10:23:05	10:23:00	9.17	Booster
2005.10.6	17:39:09	17:39:01	9.49	Booster
2005.10.7	7:35:08	7:35:01	9.32	Booster
2005.10.7	16:23:07	16:23:00	9.45	Booster
2005.10.7	17:40:08	17:40:00	9.61	Booster
2005.10.7	23:58:09	23:58:02	1.92	Booster
2005.10.8	12:45:07	12:45:00	1.52	Booster
2005.10.8	13:20:13	13:20:00	1.91	Booster
2005.10.8	13:58:13	13:58:01	2.13	Booster
2005.10.8	15:09:09	15:09:00	2.17	Booster
2005.10.9	21:22:09	21:22:00	2.13	Booster
2005.10.9	21:51:10	21:51:00	2.08	Booster
2005.10.10	11:27:08	11:27:00	1.73	Booster
2005.10.12	1:57:10	1:57:00	2.05	Booster
2005.10.12	11:55:10	11:55:00	1.83	Booster

Table 4.6 Evaluation of the injected charge of the burst current at 13th November 2004 by using the linac BPM data. Two beam slits installed in the L3BT and the L4BT almost cut the burst current. The investigation result proved that this injection of the burst current into NewSUBARU didn't become an issue of radiation safety control.

Date and time	Charge at downstream of the ECS [nC]	Charge at L3BT [nC]	Charge at the downstream of the L4BT beam slit [nC]
2004.11.13 2:51:52	7.4 ~ 7.8	1.3 ~ 1.5	0.28 ~ 0.46

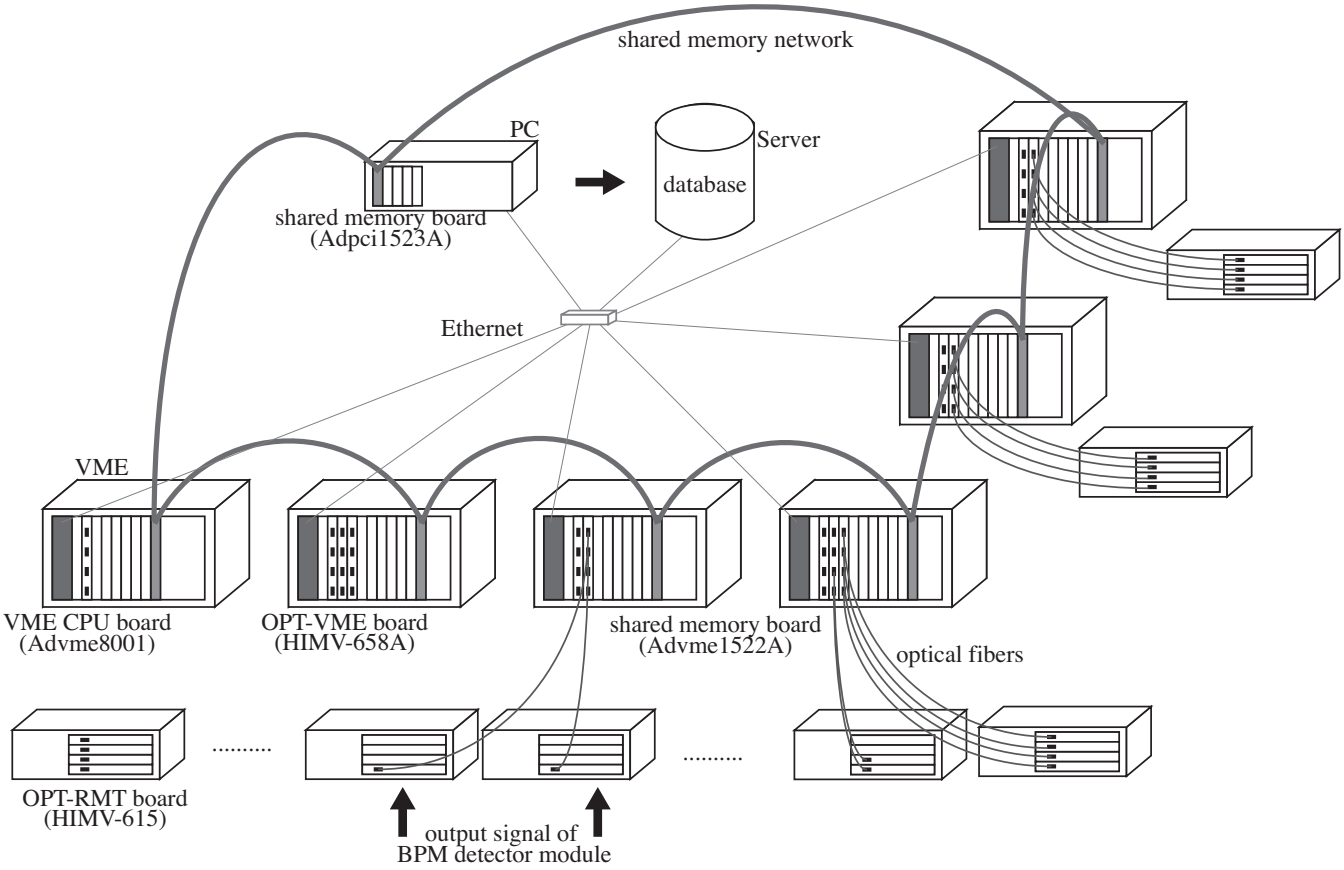


Fig. 4.1. Hardware configuration of the event-synchronized data acquisition system for the linac BPMs. The system consists of six VME computers, a PC and a database server. Each VME system has one shared memory board and some OPT-VME masters. Each OPT-VME master board controls up to four OPT-RMT boards. Each OPT-RMT board reads the digital output of one BPM detector module: (16bits + 1bit) × 4 sets.

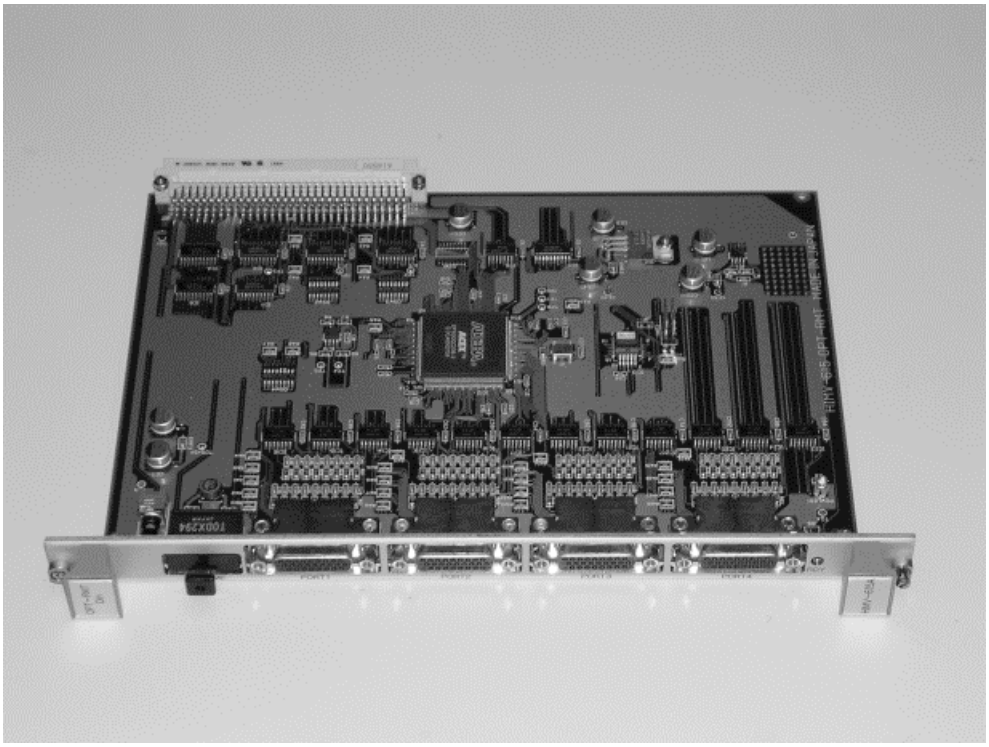


Fig. 4.2. A picture of the newly developed HIMV-615 remote board. The HIMV-615 has 4 ports of 16-bit TTL-level digital inputs with one strobe signal. Each port interfaces with one channel of a BPM detector module via a D-SUB 26 pin connector on the front panel. The HIMV-615 also has a VMEbus J1 interface for debugging.

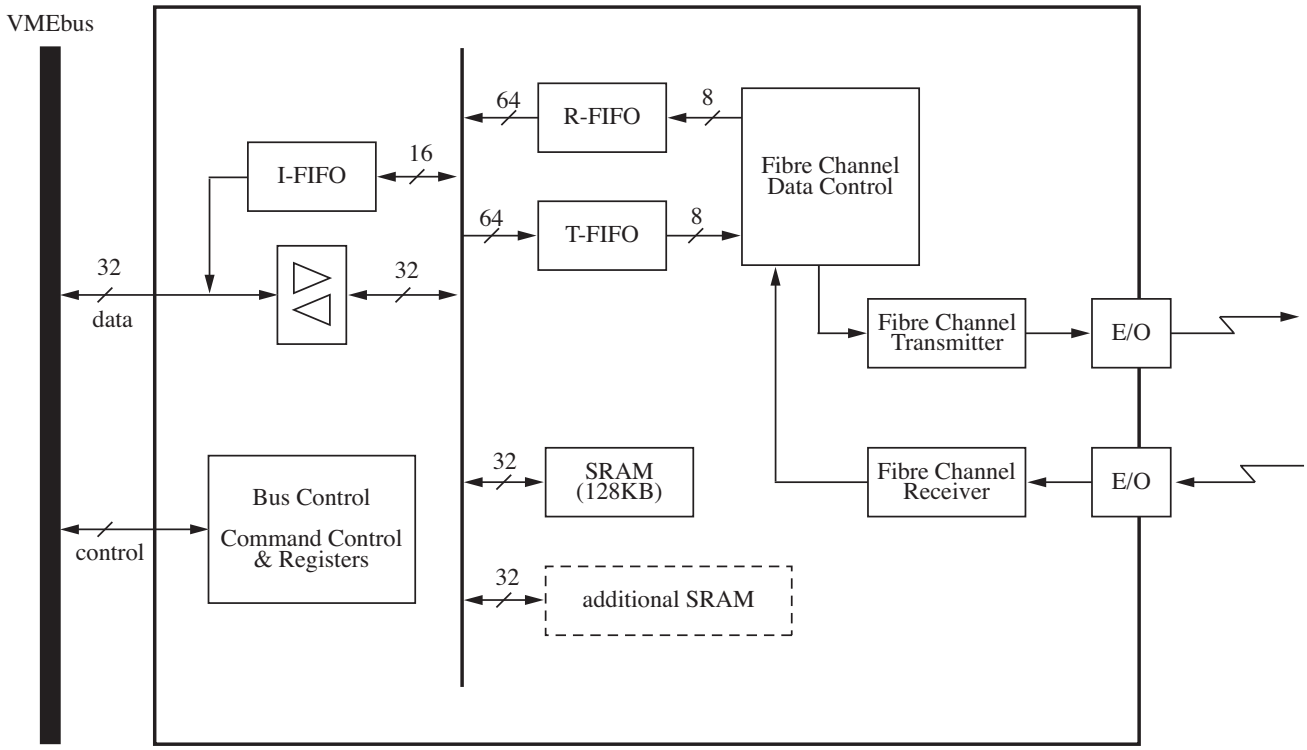


Fig. 4.3. Block diagram of the shared memory board, Advme1522A. There are three types of FIFO on the board, i.e. T-FIFO, R-FIFO and I-FIFO. Each FIFO has a depth of 512 data frames. T-FIFO (R-FIFO) is used for data transmission (reception) to (from) the SHM-net, respectively. I-FIFO is employed for the interrupt command on the SHM-net and used to interrupt the host CPU on the VMEbus.

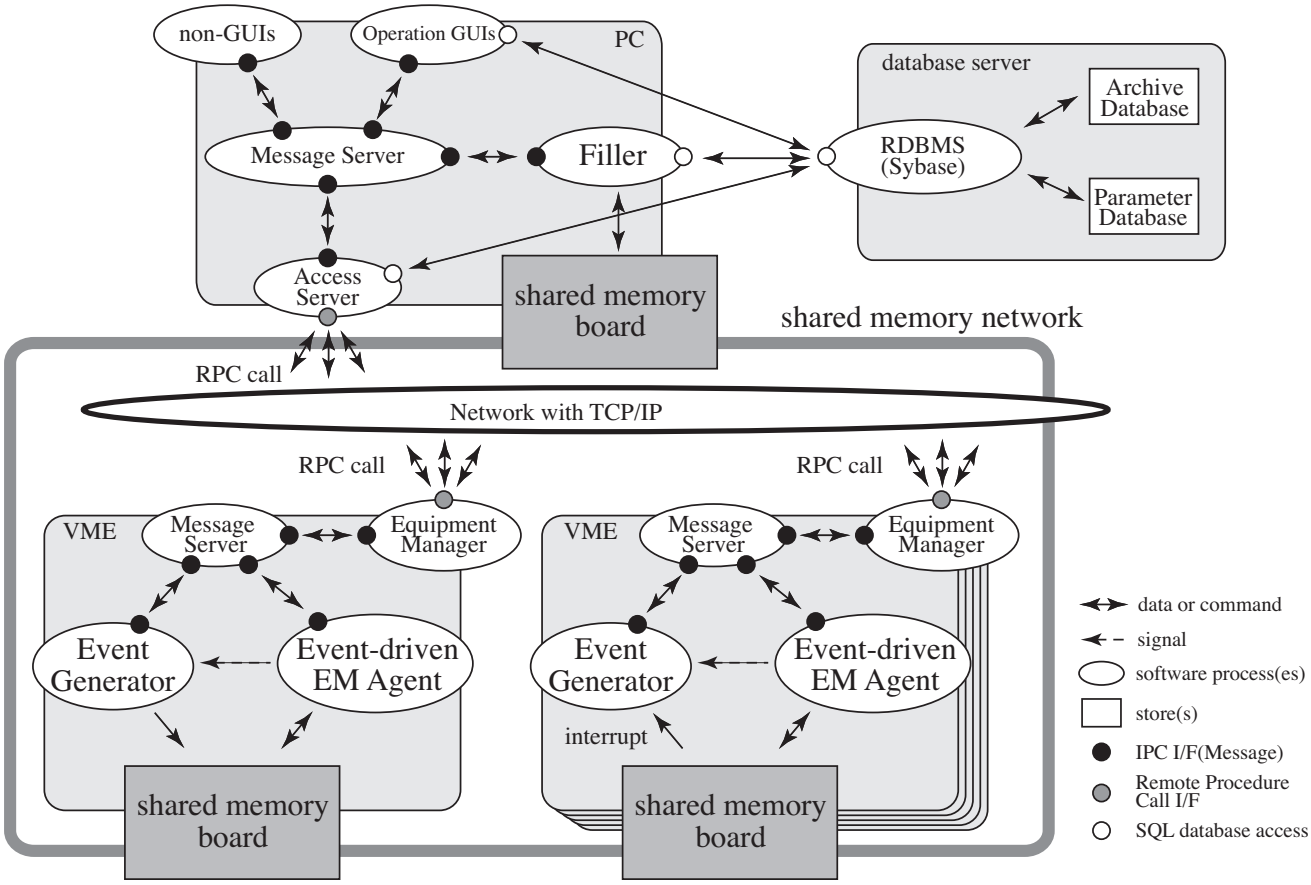


Fig. 4.4. Schematic diagram of the newly developed software framework for event-driven data acquisition based on the MADOCA framework. The EMA was modified as event-driven data acquisition software (EMA-EV), and the EVGEN was added to generate a software event. An event is signaled by using the IPC message provided by SystemV UNIX. The EMA-EV controls the EVGEN by UNIX signals. The Filler process collects all the updated data from the shared memory network and inserts them into the database.

<project>id=1,name=limonbpm</project>
<filler>id=9001,name=libpmfiller1,hostname=libpmcnt,rngsize=600,shmbdid=0,file=/dev/shm_adpci1523_0</filler>
<emaev>id=9002,name=li_mon_bpm_h0_emaev_0,hostname=libpmh0,shmbdid=16,fillerid=9001,master=1,exec=emaev_1_sol</emaev>
<signal>id=900201,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/voltage1</signal>
<signal>id=900202,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/voltage2</signal>
<signal>id=900203,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/voltage3</signal>
<signal>id=900204,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/voltage4</signal>
<signal>id=900205,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/posx</signal>
<signal>id=900206,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/posy</signal>
<signal>id=900207,emaevid=9002,type=float,kind=1,signame=li_mon_bpm_h0_1/err</signal>
…….
<emaev>id=9003,name=li_mon_bpm_h5_emaev_0,hostname=libpmh5,shmbdid=17,fillerid=9001,master=0,exec=emaev_1_sol</emaev>
<signal>id=900301,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/voltage1</signal>
<signal>id=900302,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/voltage2</signal>
<signal>id=900303,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/voltage3</signal>
<signal>id=900304,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/voltage4</signal>
<signal>id=900305,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/posx</signal>
<signal>id=900306,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/posy</signal>
<signal>id=900307,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/err</signal>
<signal>id=900308,emaevid=9003,type=float,kind=1,signame=li_mon_bpm_h1_3/average</signal>
…….
…….
<emaev>id=9007,name=li_mon_bpm_l4bt_emaev_0,hostname=libpml4bt,shmbdid=21,fillerid=9001,master=0,exec=emaev_1_sol</emaev>
<signal>id=900701,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/voltage1</signal>
<signal>id=900702,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/voltage2</signal>
<signal>id=900703,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/voltage3</signal>
<signal>id=900704,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/voltage4</signal>
<signal>id=900705,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/posx</signal>
<signal>id=900706,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/posy</signal>
<signal>id=900707,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/err</signal>
<signal>id=900708,emaevid=9007,type=float,kind=1,signame=li_mon_bpm_l4bt_3/average</signal>
…….

Fig. 4.5. A part of a *project file*. There is one <project></project> tag, one <filler></filler> tag, three <emaev></emaev> tags and some <signal></signal> tags. From the <project></project> tag, it is seen that the *project* is named *libpmmon* and assigned an ID number of *1*. The <filler></filler> tag shows the Filler has a name of *libpmfiller1* and an ID number of *9001*, and runs on the host *libpmcnt*. The record size of the ring buffer of the shared memory network is *600*. From the first <emaev></emaev> tag in the *project file*, the EMA-EV named *li_mon_bpm_h0_emaev_0* has an ID number of *9002*, and is controlled by the Filler whose ID number is *9001*. The EMA-EV runs on a host *libpmh0* as an executable filename *emaev_1_sol*. This EMA-EV generates a master EVGEN as a child process because master qualification is set to *1*. From the first <signal></signal> tag, a signal of *li_mon_bpm_h0_1/voltage1* has an ID number *900201*, and is collected by an EMA-EV with an ID number *9002*.

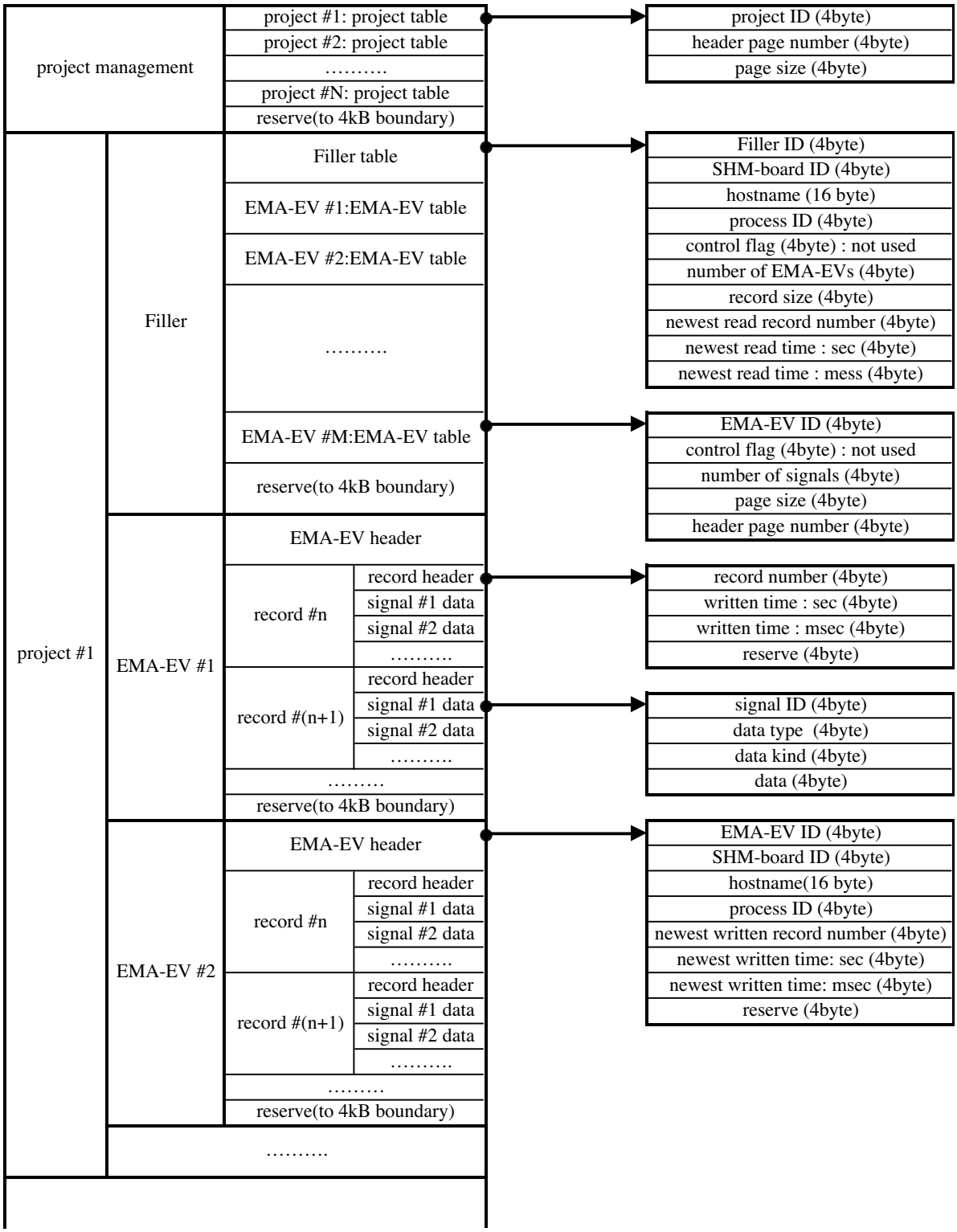
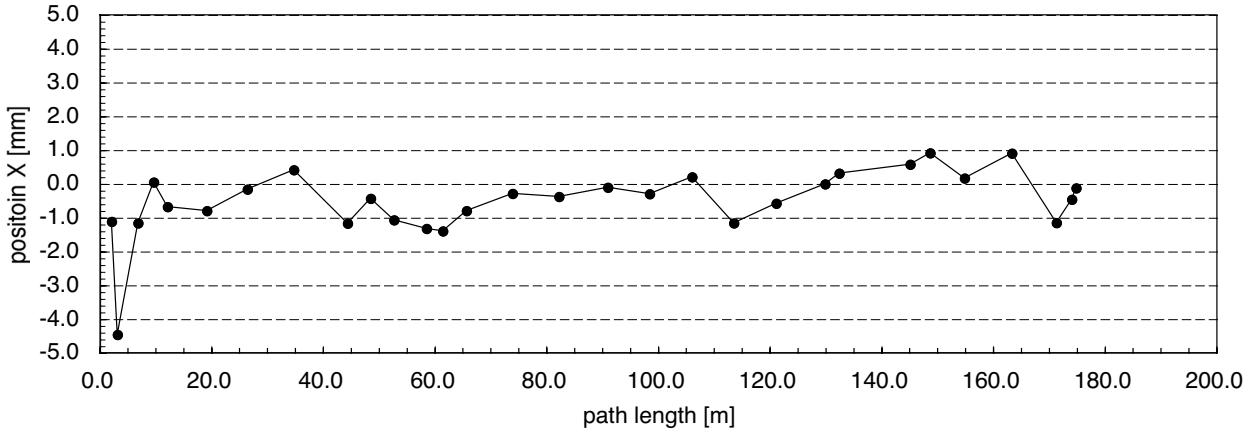


Fig. 4.6. Structure of the data table of the *project* on the shared memory network.

(a)



(b)

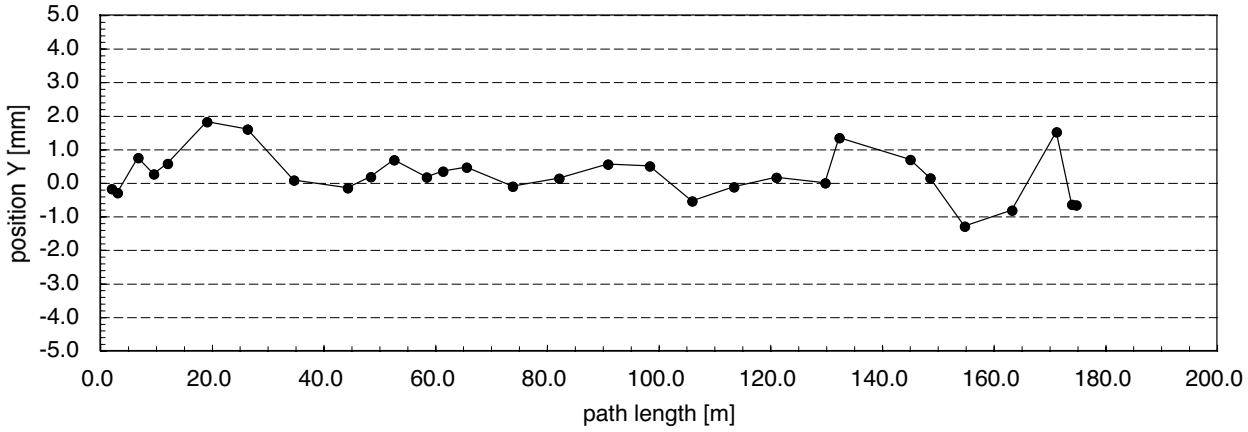


Fig. 4.7. Beam positions of one beam shot as collected by all the linac BPMs. Graph (a) shows the horizontal beam positions and graph (b) shows the vertical beam positions. The path length is measured in distance from the electron gun.

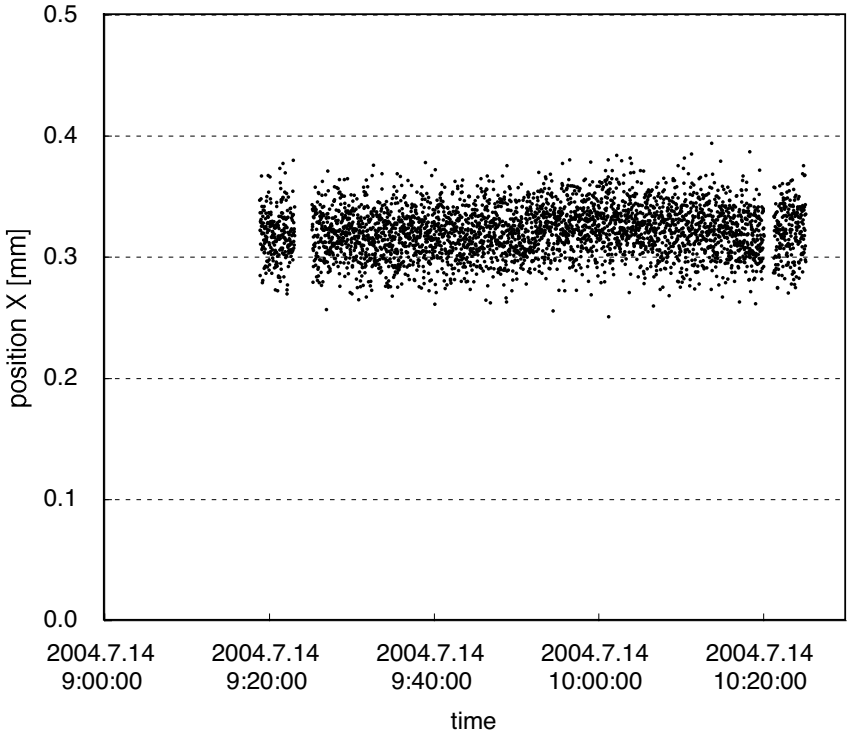
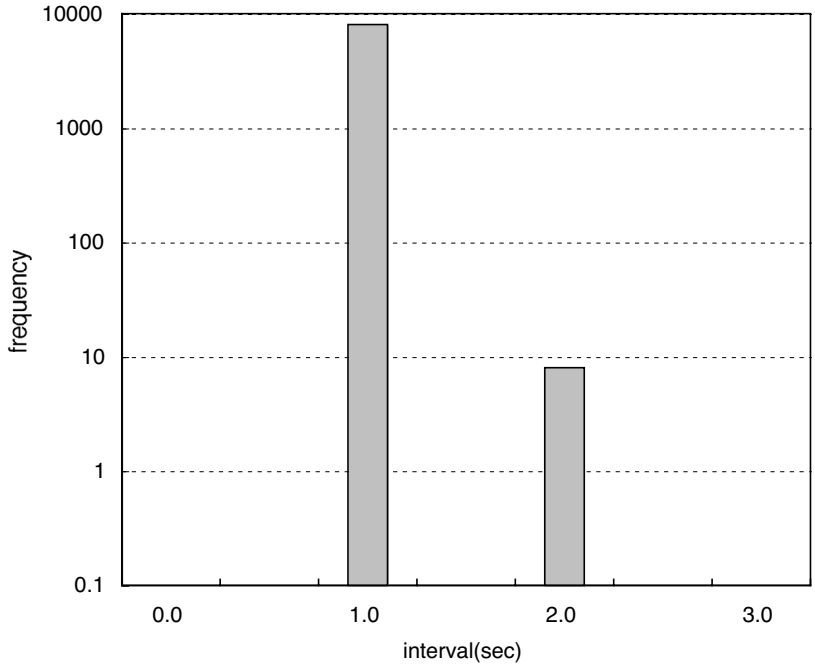


Fig. 4.8. An example of time variation of horizontal beam position at one BPM. These data were collected during accelerator tuning, and thus a stable beam was not expected. But the result indicates the horizontal beam positions were stable at this BPM position.

(a)



(b)

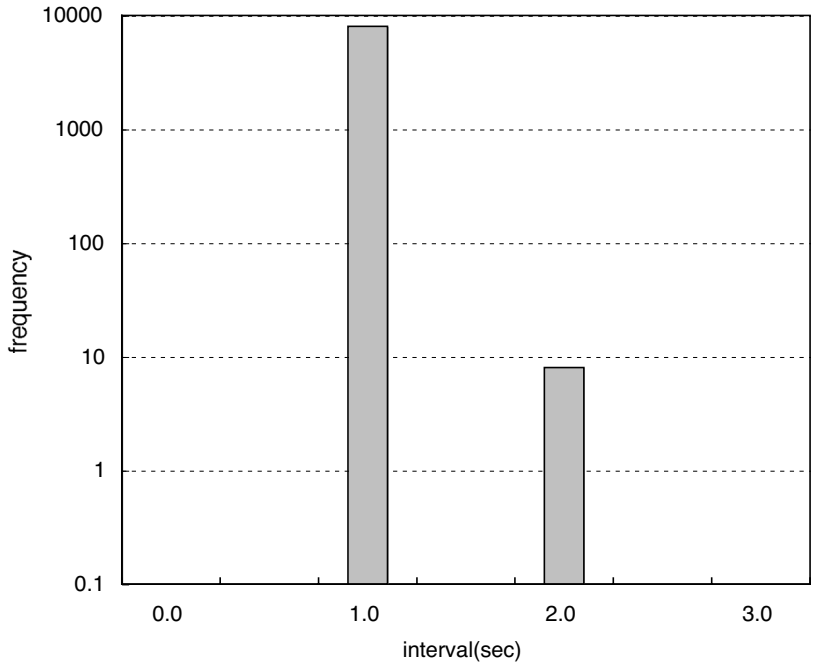
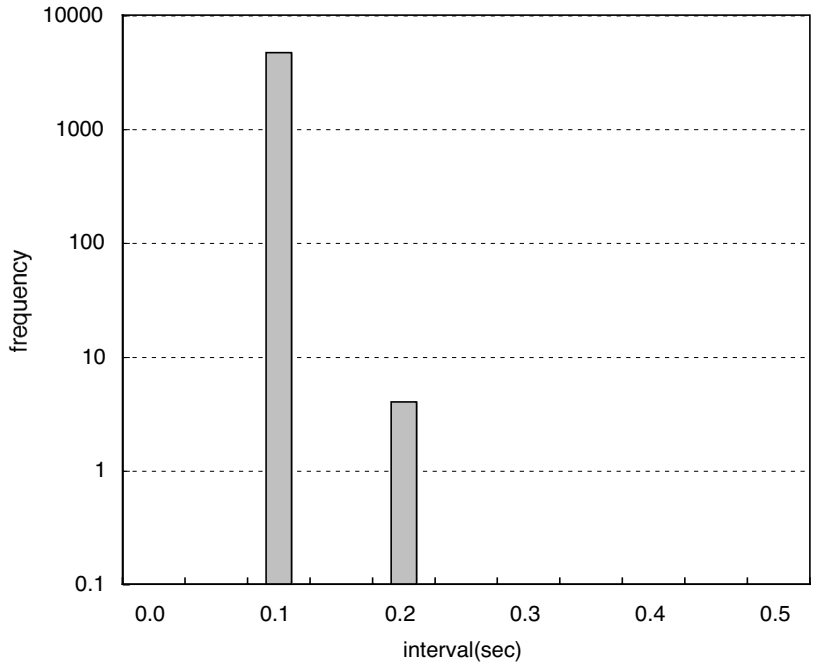


Fig. 4.9. Histograms of data acquisition interval between consecutive captured events for 1pps linac operation. Fig. (a) shows the result for host libpmm20, which acquires the most numbers of the BPM signals. Fig. (b) shows the result for host libpmh0, which employs the master EVGEN. Both figures show the same number of lost events.

(a)



(b)

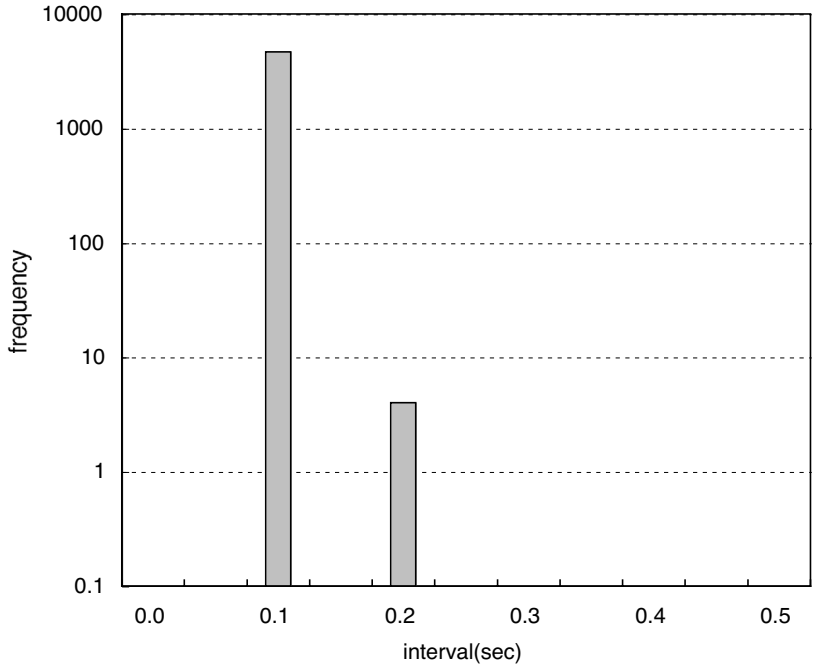
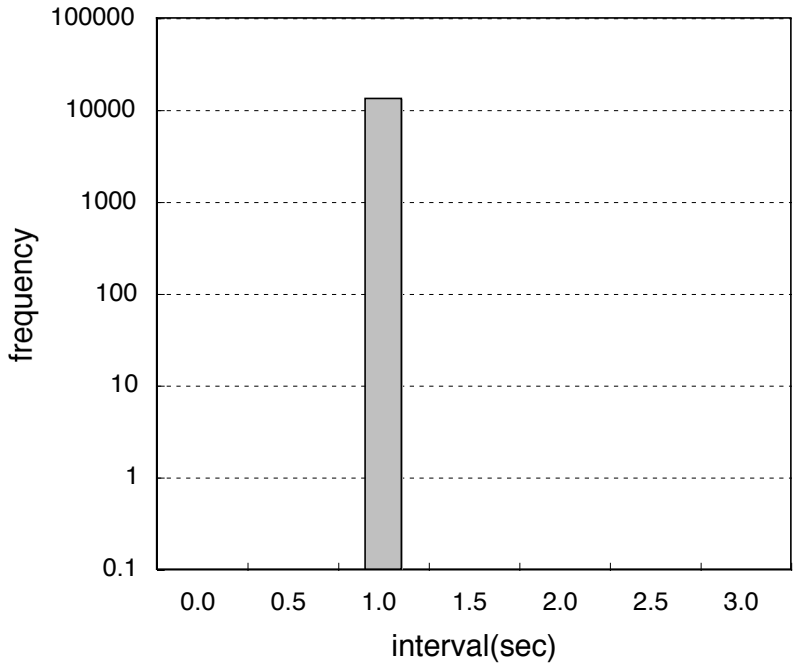


Fig. 4.10. Histograms of data acquisition interval between consecutive captured events for 10pps linac operation. Fig. (a) shows the collection result for host libpmm20, and the Fig. (b) shows the result for host libpmh0. Figs. (a) and (b) display the same result.

(a)



(b)

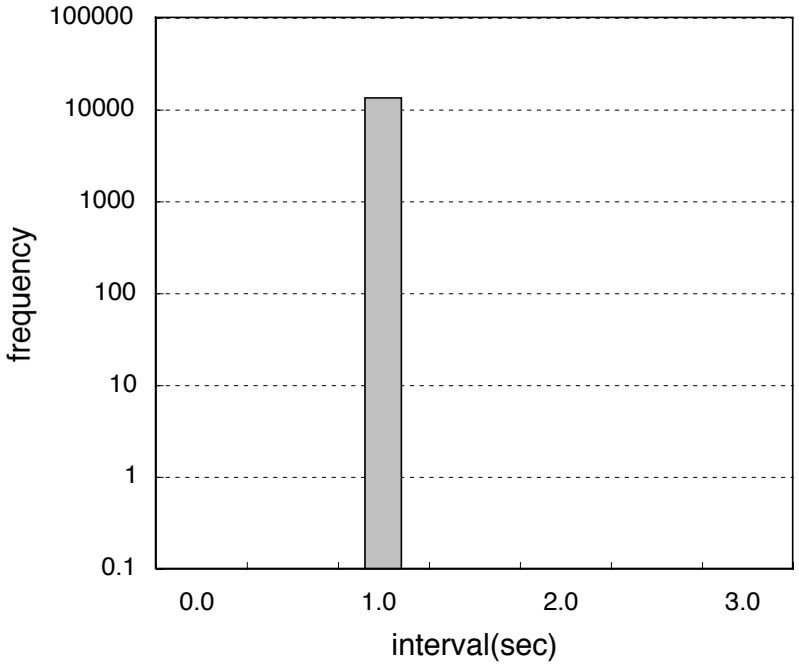
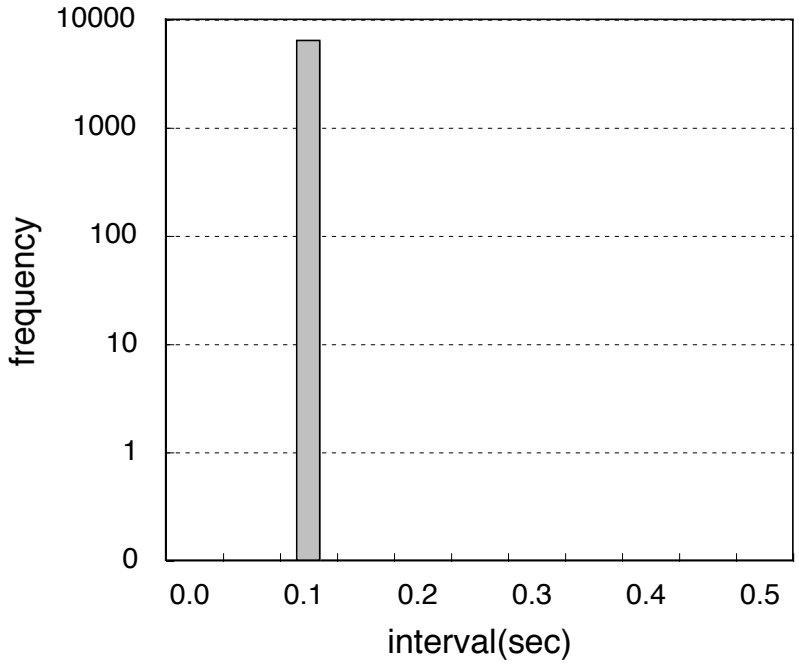


Fig. 4.11. Histograms of data acquisition interval between consecutive captured events for 1pps linac operation by using the modified data acquisition system. Fig. (a) shows the result for host libpmm20 and Fig. (b) shows the result for host libpmh0. The improved data acquisition system succeeded in taking all the events.

(a)



(b)

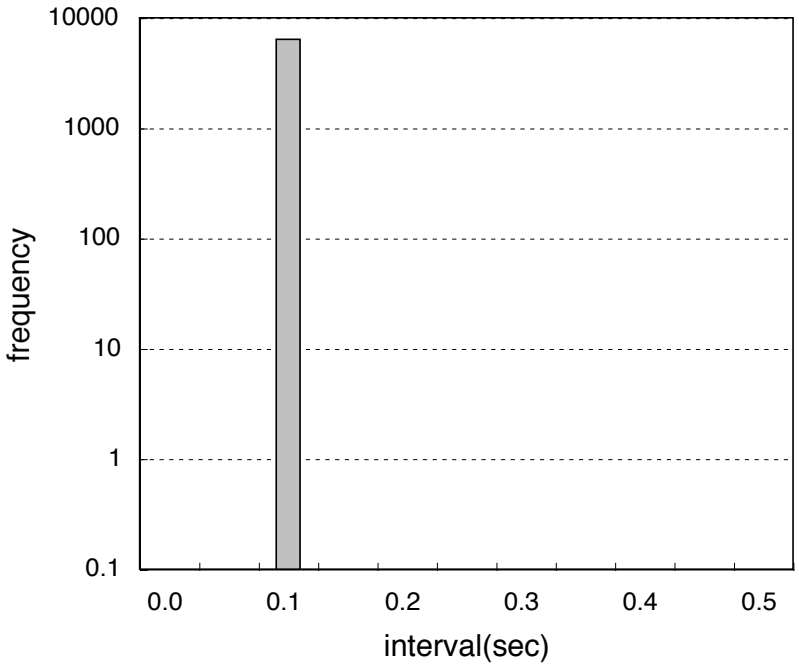
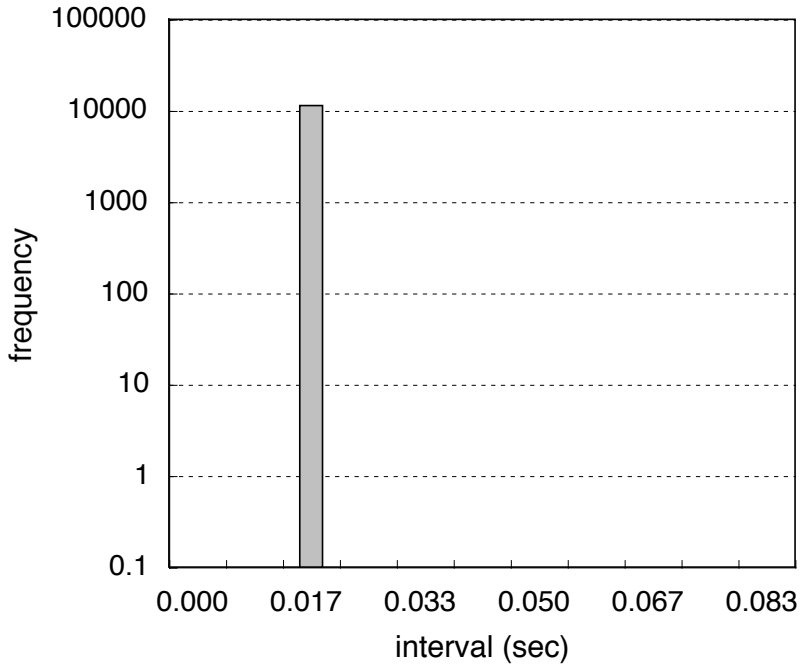


Fig. 4.12. Histograms of data acquisition interval between consecutive captured events for 10pps linac operation by using the modified data acquisition system. Fig. (a) shows the collection result for host libpmm20, and the Fig. (b) shows the result for host libpmh0. The data acquisition system successfully collected all the 10Hz events with no losses.

(a)



(b)

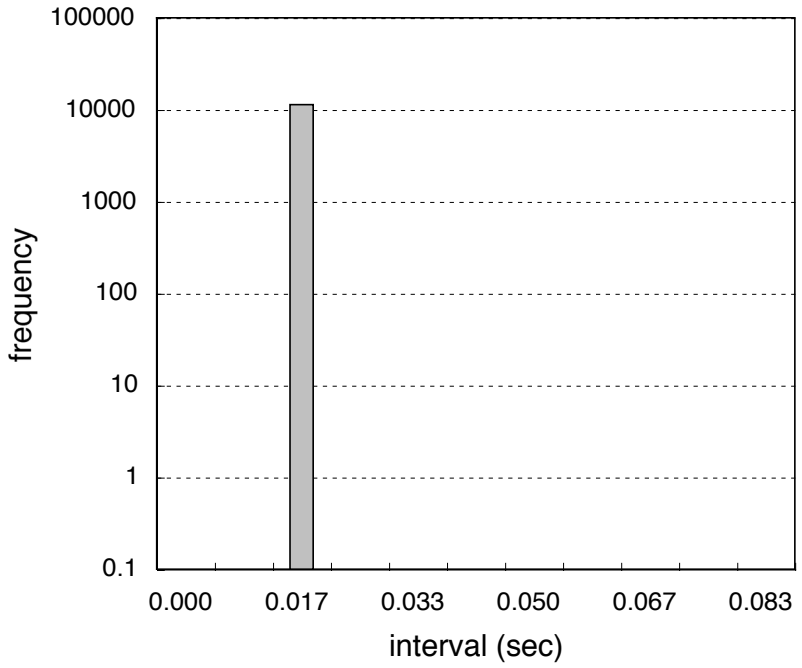
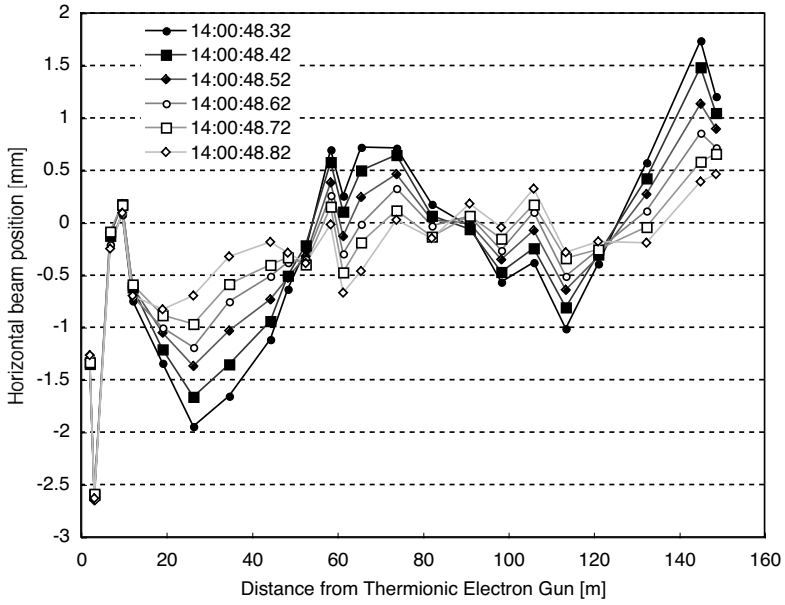


Fig. 4.13. Histograms of data acquisition interval between consecutive captured events for intermittent 60pps linac operation by using the modified data acquisition system. Fig. (a) shows the collection result for host libpmm20, and the Fig. (b) shows the result for host libpmh0. The data acquisition system successfully collected all the 60Hz events with no losses.

(a)



(b)

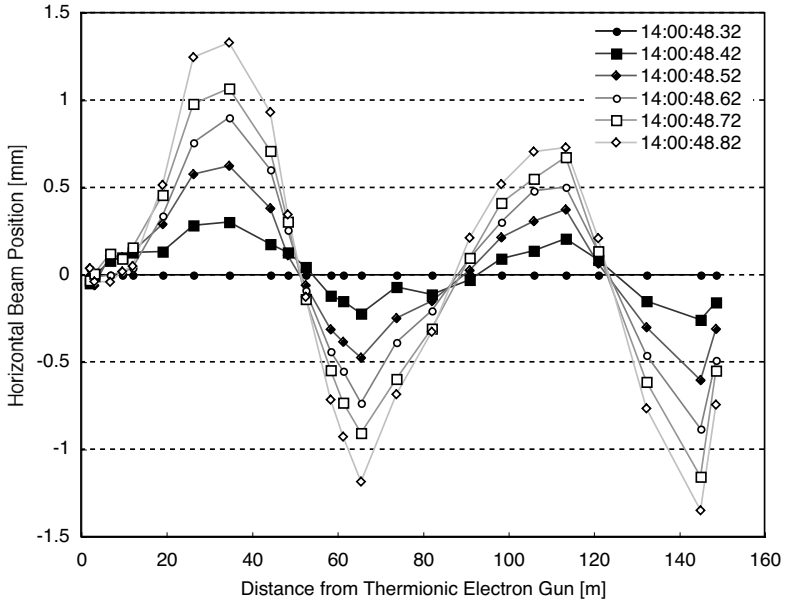


Fig. 4.14. Horizontal beam trajectories (in Fig. (a)) and their variations (in Fig. (b)) along the length of the linac at 10pps operation. The variations are expressed as the differences from the trajectory at 14:00:48.32. They clearly show betatron oscillations in the linac.

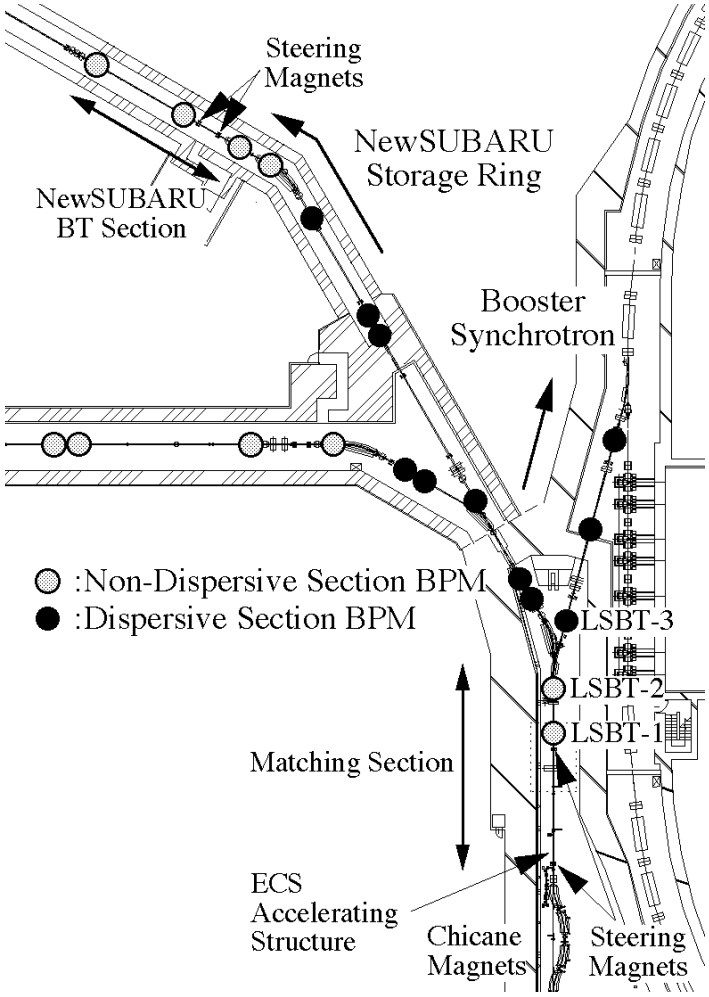
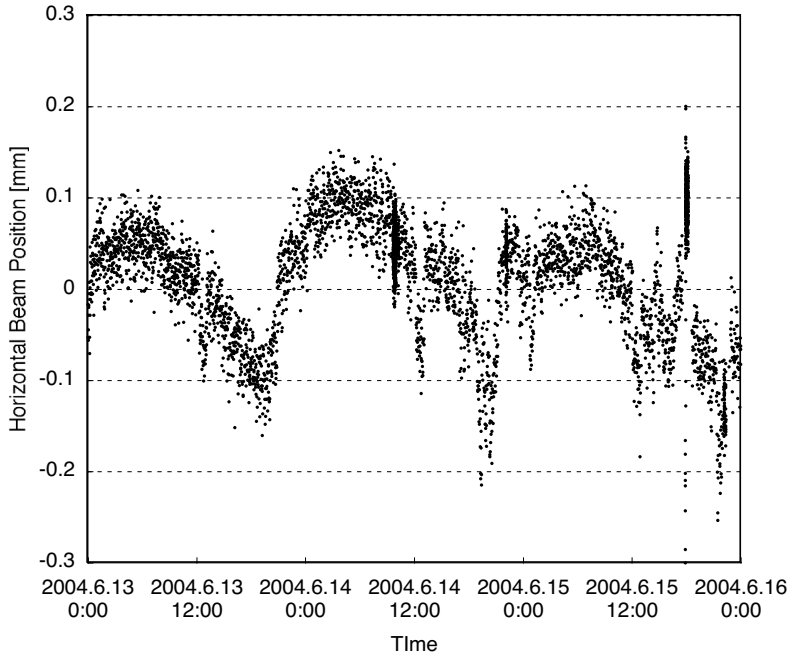


Fig. 4.15. The matching section located at the downstream of the ECS system and the BT section to the NewSUBARU. There are two sets of steering magnets in the drift space located at the upstream of the two sets of BPMs in the non-dispersive section. Beam positions and angles to the booster are finally stabilized at the matching section, and those to the NewSUBARU are finally stabilized at the BT section.

(a)



(b)

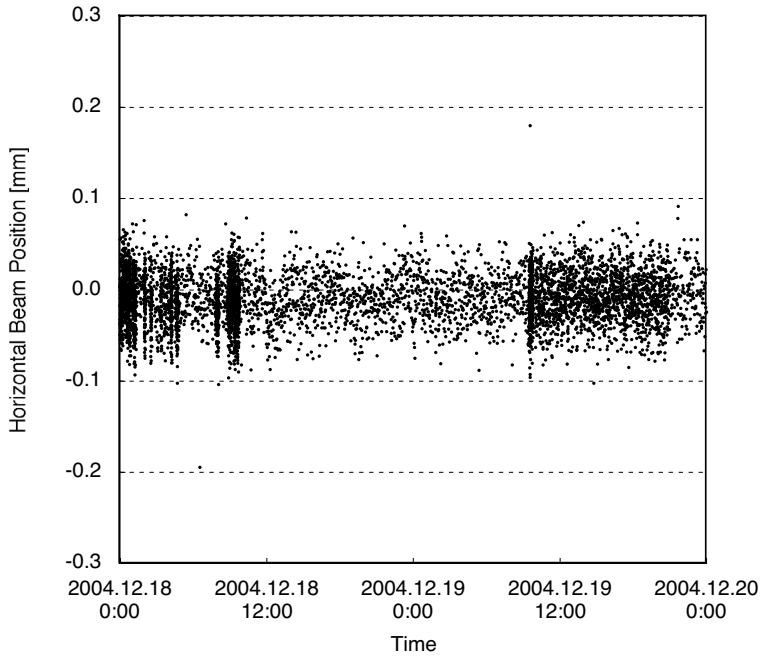
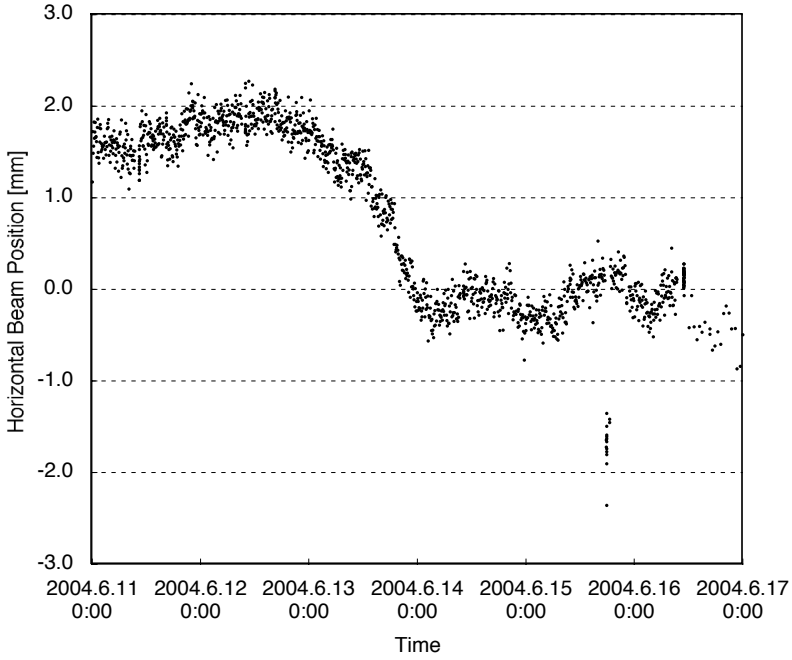


Fig. 4.16. Fig. (a) shows time variation of horizontal beam position at the LSBT-1 BPM in the matching section before introducing an automatic beam position feedback control. And Fig. (b) shows the result of the introducing the position feedback control at the same BPM. From their comparison, the automatic position feedback control succeeds in stabilization of the horizontal beam position at the matching section.

(a)



(b)

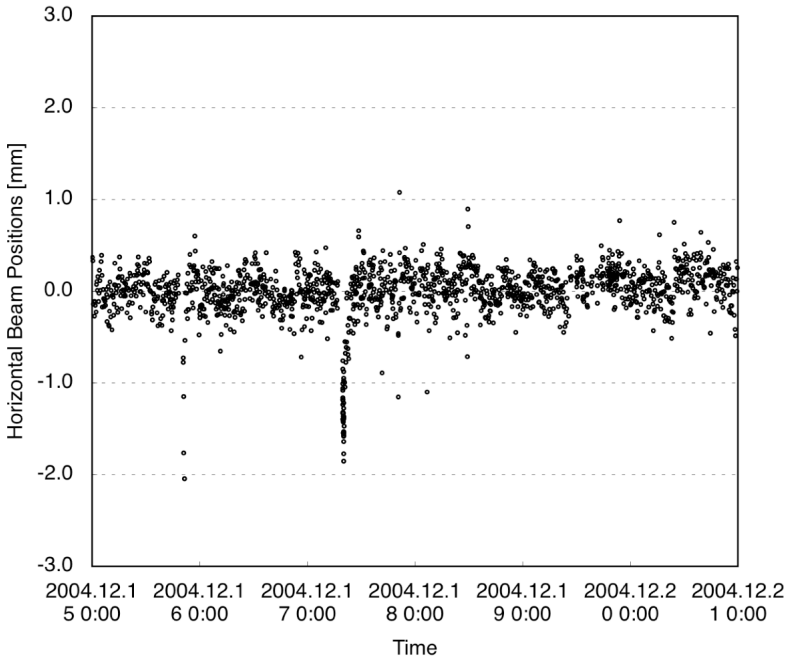


Fig. 4.17. Fig. (a) shows time variation of horizontal beam position at the LSBT-3 BPM before introducing an automatic beam energy feedback control. And Fig. (b) shows the result of the introducing the energy feedback control at the same BPM. From their comparison, the automatic energy feedback control succeeds in stabilization of the beam energy.

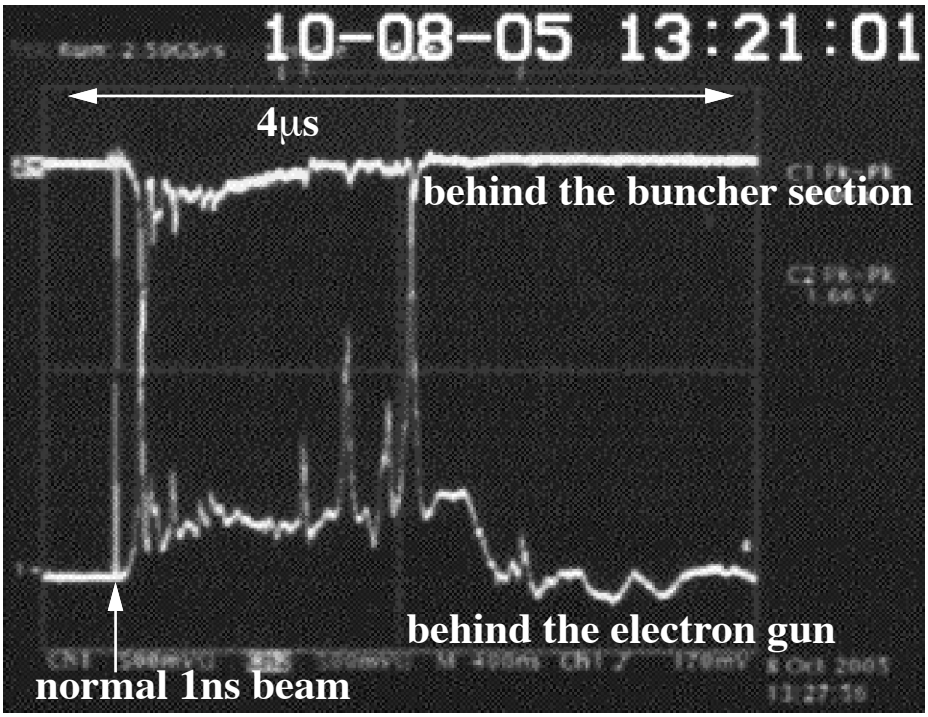


Fig. 4.18. The burst current observed using the beam current monitors. Lower line (upper line) is observed by the current monitor installed just behind the electron gun (the buncher section), respectively. The burst current was emitted 40nsec after the normal 1nsec beam with the length of 2msec. The total charge of the burst current was nearly 10nC.

5. Summaries and conclusions

In this study, development of the linac control system toward stable linac operation was presented. The results of this study are summarized and concluded as follows.

The new linac control system was successfully developed taking account of stability enhancement of the control system itself and availability enhancement of the linac operation. The MADOCA framework was applied for the control software. It newly brought enough stability, powerful data logging/archiving system and sufficient throughput of issued control commands to the linac accelerator equipment. For applying MADOCA, Solaris was newly chosen as the real-time OS for the Intel architecture VME CPU boards, according to the results of wide and profound studies of real-time functionality. The data logging/archiving system greatly contributed to the linac stabilization by providing the off-line analyses capability. The control hardware was radically renovated. The newly developed OPT-VME and MCU replaced direct I/O boards on the VMEbus. Adopting these new devices successfully brought noise immunity enhancement, function-oriented control system re-arrangement and availability enhancement of the linac operation.

In 2005, the VME systems were rebooted seven times. All the troubles were due to the UPD software problem. Nevertheless, these troubles had never interrupted the top-up operation of SPring-8. The fact showed that the approach for the availability enhancement in this study was considerably successful.

On the basis of the new linac control system, the shot-by-shot data acquisition system for the linac BPM was newly developed. The data acquisition system successfully took 376 signals from 47 BPMs synchronizing to the 60pps linac operation without any shot detection losses. Six VME computers, a PC and a dedicated database server were used for the acquisition. The SHM-net linked all the VME computers and the PC for fast real-time transfer of the acquired BPM data. The SHM-net was also used for synchronization between software processes running on the VME computers. A set of synchronized BPM data was recorded into a commercial RDB on the dedicated database server together with an event number and a time stamp for beam shot identification. All the acquired BPM data were kept permanently in the RDB and could be easily accessed by using web browsers or C function calls.

The data acquisition system presently needed about 50msec for a SQL data insertion into the RDB. The ring-buffered data bank built on the SHM-net helped the system to acquire all the data at 60Hz operation as long as the ring-buffer didn't overflow. The system has the potential to achieve continuous 60Hz data acquisition soon by using bulk insert technique, which already achieved 100Hz data insertions of 400 signals into a DB in the test.

The accumulated BPM data successfully gave opportunities of precise analyses of beam trajectories in the linac and provided capability of feedback control of the linac electron beams to achieve long-term energy

stability. Before introducing the automatic corrections of the electron beams trajectories and energies in every 5 minutes, operators had manually corrected them whenever the injection efficiency had become worse. The BPM data analysis successfully contributed to the achievement of long-term energy stability of 0.02% (rms) and the beam positions stability of 30μm (rms) through the automatic feedback control.

The longtime accumulated BPM data enables reproduction of a past electron beam status. The analysis of the burst currents, which occurred very occasionally in the linac, is a good example to show the importance of the completeness of BPM dataset. An investigation of recorded BPM data brought a fruitful result to find a rare burst-current phenomenon. It was achieved by surveying a perfect dataset taken by the event-synchronized data acquisition system.

This study showed that the beam-synchronized data acquisition and the analysis of all the acquired data was effective approach for the linac stabilization. Since the BPM data acquisition system was developed as the standard framework of MADOCA for the event-synchronized data acquisition, the new framework, furthermore, can be applied to 60Hz data acquisition of the SCSS 8-GeV XFEL accelerator whose control system will be developed with MADOCA. The application will help the beam stabilization of SCSS a lot.

The approach in this study to stable linac operation is general and widely adaptable for other linac control system, which doesn't use the MADOCA framework. The approach to hold present output signals without depending on any states of equipment control computers is very effective to availability enhancement of a linac operation. And, data acquisition synchronizing to every beam shot and accumulation of all the acquired data into a general RDB is effective approach for stabilization of a linac. A beam-shot synchronized complete BPM dataset can be used for feedback control and feed-forward control to stabilize linac beams. Furthermore, the accumulated datasets plays an essential role to reproduce past electron beam status, providing opportunities for deeper understanding of beam phenomena to achieve ultimate beam stabilization. In particular, the new system developed in this study is indispensable for 4th-generation 8-GeV XFEL being built in SPring-8 to produce X-ray laser of wavelength 0.06nm.

Acknowledgements

I would like to express sincere gratitude to Dr. Noritaka Kumagai for his encouragement and support to give me an opportunity of this work. He also gave me invaluable advices to complete this thesis.

I would like to express my great gratitude to Prof. Shin-ichi Kurokawa at KEK for his kind and invaluable advices to complete this thesis. I wish to express my gratitude to Professors Setsuya Kawabata, Hirofumi Fujii, Junji Urakawa and Toshio Kasuga at KEK for their invaluable comments.

I would like to express my profound thanks to Dr. Ryotaro Tanaka for his many invaluable suggestions and advices. I couldn't accomplish this work without helpful discussion with him and his continuous encouragement throughout this work.

I wish to thank Dr. Akihiro Yamashita for his useful advices and his help of databases. I am grateful to Dr. Toru Fukui for his many suggestions and advices on the linac control system renovation and newly developed data acquisition system. I express my thanks to Mr. Kenichi Yanagida for his useful advices and cooperation in development of the MCU, the connector box, and BPM data acquisition system. I would like to thank Dr. Toru Ohata for helpful discussion and collaboration in the studies of the VME CPU board and OS.

I am grateful to Drs. Naoyasu Hosoda, Tsutomu Taniuchi, and Messrs. Masahiko Kodera, Kazuo Kobayashi, Ms. Miho Ishii, and Mrs. Yukiko Taniuchi for their advices and cooperation in the work of the linac control system renovation. I also thank Drs. Hirofumi Hanaki, Toshiaki Kobayashi, Shinsuke Suzuki, Hideki Dewa, Takao Asaka, Hiromitsu Tomizawa, and Mr. Akihiko Mizuno for their help in the work of the linac control system renovation.

I would like to appreciate Mr. Roger Bissonnete for his valuable advices, and careful reading of my contribution paper.

I would like to express my great to thanks Drs. Hitoshi Tanaka and Mitsuhiro Masaki for their advice and encouragement.

I am most grateful to Prof. Yoshihiro Tagishi at University of Tsukuba for his much helpful guidance and continuous encouragement.

I am always grateful to my parents and my elder sister for their continuous encouragement.

At last but not the least, I would like to express my sincere thanks to my wife, Mayumi, for her many supports and continuous encouragement.

References

[1] H. Kamitsubo, Nucl. Instrum. Methods. A 303 (1991) 421-434.

[2] H. Tanaka *et al.*, "Top-up Operation at SPring-8 – Towards Maximizing the Potential of a 3r Generation Light Source", Proc. of EPAC'04, Lucerne, Switzerland, 2004, p. 222.

[3] A. Ando *et al.*, "Isochronous storage ring of the NewSUBARU project", J. Synchrotron Rad. 5 (1998) pp.342-344.

[4] T. Asaka, Doctor Thesis, Univ. Kyoto (2004).

[5] H. Hanaki, J. Particle Accelerator Society of Japan, Vol2, No.3, 2005, 317-329 (in Japanese)

[6] T. Asaka *et al.*, "Performance of the Energy Compression System at the SPring-8 Linac", Proc. of EPAC'02, Paris, France, 2002, p. 1079.

[7] T. Asaka *et al.*, Nucl. Instrum. Methods. A 516 (2004) 249-269.

[8] Y. Kawashima *et al.*, Phy. Rev. Special Top. Accelerator Beams 4, 082001 (2001).

[9] R. Tanaka et al., "The first operation of control system at the SPring-8 storage ring", Proc. of ICALEPCS'97, Beijing, China, 1997, p. 1.

[10] http://www-xfel.spring8.or.jp

[11] T. Masuda *et al.*, "Upgrade of Linac Control System with New VME Controllers at SPring-8", Proc. of ICALEPCS'01, San Jose, USA, 2001, p 228.

[12] K. Yanagida et al., "A BPM System for the SPring-8 Linac", Proc. of the 20th Int. Linac Conf., Monterey, USA, 2000, p. 190.

[13] J. A. Carwardine *et al.*, "Architecture of the APS Real-Time Orbit Feedback System", Proc. of ICALEPCS'97, Beijing, China, 1997, p. 470.

[14] C. H. Kuo *et al.*, "Upgrading the Orbit Feedback System in the Taiwan Light Source", Proc. of PAC'03, Portland, USA, 2003, p. 3392.

[15] S. Suzuki *et at.*, "Improvements of SPring-8 Linac Towards Top-up Operation", Proc. of EPAC'04, Lucerne, Switzerland, 2004, p.1327

[16] T. Asaka *et al.*, "Improvement of the Injector Linac for Top-up Operation at the SPring-8", Proc. of 29th Linear Accelerator Meeting in Japan, Funabashi, Japan, 2004, p489 (in Japanese)

[17] K. Yanagida *et al.*, "A BPM System for the SPring-8 Linac", Proc. of the 20th Int. Linac Conf., Monterey, USA, 2000, p. 190.

[18] K. Yanagida *et al.*, "Installation of the SPring-8 Linac BPM system", Proc. of LINAC2002, Gyeongju, Korea, 2002, p.448

[19] K. Yanagida *et al.*, "Signal Processor for SPring-8 Linac BPM", Proc. of DIPAC 2001, Grenoble, France, 2001, p. 162.

[20] Analog Devices, Inc., http://www.analog.com

[21] T. Wada *et al.*, “Design of SPring-8 Control System”, Proc of ICALEPCS’91, Tsukuba, Japan, 1991 p.151

[22] R. Tanaka *et al.*, “Control System of the SPring-8 Storage Ring”, Proc of ICALEPCS’95, Chicago, USA, 1995 p.201

[23] Hewlett-Packard Development Company, http://www.hp.com

[24] T. Masuda *et al.*, “Development of Device Drivers on Real-time UNIX for the SPring-8 Storage Ring Control System”, Proc of ICALEPCS’95, Chicago, USA, 1995 p.584

[25] Lynuxworks, Inc., http://www.lynuxworks.com

[26] Sun Microsystems, http://www.sun.com

[27] Mitsubishi Electric Co., http://www.mitsubishielectric.co.jp

[28] H. Takebe *et al.*, “Magnet Power Supply Control System for the SPring-8 Storage Ring - Fiber distributed Remote I/O System for VME ”, Proc. of EPAC’94, London, UK, 1994, p1827

[29] T. Masuda *et al.*, “Development of PC based Field Controller with Linux Operating System”, Proc. of PCaPAC’99, Tsukuba, Japan, 1999.

[30] National Instruments corporation, http://www.ni.com

[31] Sybase Inc., http://www.sybase.com

[32] Red Hat, Inc., http://www.redhat.com

[33] T. Fukui *et al.*, “Design and performance of the network system for the storage ring control at SPring-8”, Proc of ICALEPCS'97, Beijing, China, 1997 p.312.

[34] T. Fukui *et al.*, “Toward a Reliable Gigabit Network - An Upgrade of the SPring-8 Network”, Proc. of ICALEPCS'01, San Jose, USA, 2001, p.487

[35] A. Taketani *et al.*, “Equipment Manager of the VME Control System for the SPring-8 Storage Ring”, Proc of ICALEPCS’95, Chicago, USA, 1995 p.625

[36] A. Taketani *et al.*, “Medium-speed feedback software based on the existing control system”, Proc of ICALEPCS'97, Beijing, China, 1997 p.486

[37] A. Taketani *et al.*, “Data Acquisition System with Database at the SPring-8 Storage Ring”, Proc of ICALEPCS'97, Beijing, China, 1997 p.437

[38] A. Yamashita *et al.*, “The database system for the SPring-8 storage ring control”, Proc of ICALEPCS'97, Beijing, China, 1997 p.427.

[39] A. Yamashita *et al.*, “Data Archiving and Retrieval for SPring-8 Accelerator Complex”, Proc. of ICALEPCS'99, Trieste, Italy, 1999, p.434

[40] A. Yamashita *et al.*, “The Alarm System for the SPring-8 Storage Ring”, Proc of ICALEPCS'97, Beijing, China, 1997 p.585.

[41] A. Yamashita, “Distributed Database in SPring-8”, Proc. of ICALEPCS’03, Gyeongju, Korea, 2003, p.436.

[42] http://gnuplot.info

[43] Fuji Data System Co. Ltd., http://www.fujidatasystem.com (Japanese only)

[44] H. Sakaki *et al.*, "Design of SPring-8 Linac Control System Using Object Oriented Concept", Proc. of 1995 Part. Accel. Conf., Dallas, 1995.

[45] Motorola, Inc., http://www.motorola.com

[46] RadiSys Corporation, http://www.radisys.com

[47] XycomVME, Inc., http://www.xycomvme.com

[48] T. Masuda et al., "Upgrade of Linac Control System with New VME Controllers at SPring-8", Proc. of ICALEPCS'01, San Jose, USA, 2001, p.228

[49] Advanced Micro Devices, Inc., http://www.amd.com

[50] T. Ohata and T. Masuda, "A study of a real-time operating system on the Intel-based VME controllers", Proc. of ICALEPCS 2001, San Jose, USA, 2001, p 554.

[51] Nichizou Electronic & Control Corporation, http://www.ndssf.co.jp

[52] T. Fukui *et al.*, "Applications of reconfigurable logic devices for accelerator controls", Proc. of ICALEPCS'03, Gyeongju, Korea, 2003, p.241

[53] T. Masuda *et al.*, "Upgrade of the SPring-8 Linac Control by Re-engineering the VME Systems for Maxmizing Availability", Proc. of ICALEPCS'03, Gyeongju, Korea, 2003, p.295

[54] Altera Corporation, http://www.altera.com

[55] Toshiba Corporation, http://www.toshiba.co.jp

[56] Renesas Technology Corp., http://www.renesas.com

[57] MiSPO Co., Ltd., http://www.mispo.co.jp (Japanese only)

[58] ITRON project, http://www.ertl.jp/ITRON/home-e.html

[59] Interface Corporation, http://www.interface.co.jp/

[60] M. Ishii *et al.*, "A Software Framework to Control a Network-Connected Equipment as a Pseudo Device", Proc. of ICALEPCS'03, Gyeongju, Korea, 2003, p.512

[61] T. Masuda *et al.*, Nucl. Instrum. Methods. A 534 (2005) 415-430.

[62] Advanet Inc., http://www.advanet.cojp

[63] Manuals of Advme1522A board and Adpci1523A board, Advanet Inc. (in Japanese).

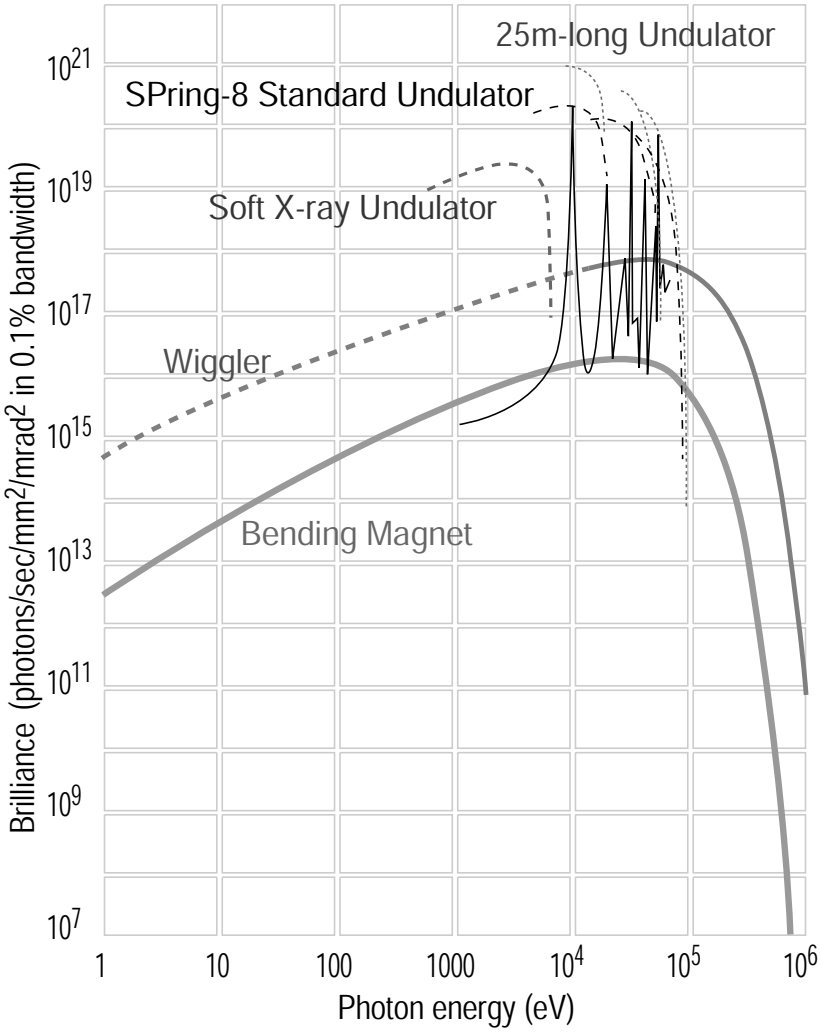
[64] K. Yanagida et al., "Beam Instrumentation using BPM system of the SPring-8 Linac", Proc. of LINAC '04, Lübeck, Germany, 2004, p. 464.

[65] Sanritz Automation Co. Ltd, http://www.sanritz.co.jp

[66] SCSS X-FEL Conceptual Design Report May 2005

Appendix

Appendix-A. Synchrotron Radiation Spectrum of the SPrnig-8



Appendix-B. Main parameters of the SPring-8 linac

Beam energy	1.2 GeV max.
Beam current	< 2 A (peak)
Beam pulse width	250ps, 1ns, 40ns
Bunch length	10 ~ 30ps (FWHM)
Normalized emittance	< 200 π mm mrad (rms)
Emittance	< 1 π mm mrad (rms)
RF frequency	2856MHz
Repetition rate	< 60
Klystron	80MW × 13
Accelerating structure	3m × 26
Electric field of accelerating structure	< 20MV/m
Input power of accelerating structure	35MW
Length of linac	140m

Appendix-C. RF parameters of the SPring-8 linac

Pre-buncher	
Type	Reentrant standing wave
Operation frequency	2856MHz
Shunt impedance	24MΩ/m
Unloaded Q	6400
Input power	7kW
Buncher	
Type	Side-couple standing wave
Operation frequency	2856MHz
Number of cells	13
Shunt impedance	103MΩ/m
Unloaded Q	13400
Input power (usual operation)	3MW
Accelerating structure	
Type	Travelling wave
Electric field distribution	Constant gradient
Phase shift / cell	2/3 π
Operation frequency	2856MHz
Shunt impedance	54MΩ/m
Unloaded Q	13600
Effective length	2.88m
Filling time	610ns
Input power (usual operation)	35MW