EPICS DEVICE CONTROLLER FOR THE BPMC OF THE J-PARC MAIN RING

G. Shen[A) #], N. Kamikubota [A)], N. Yamamoto [A)], K. Furukawa [A)], H. Nakagawa [A)], T. Katoh [A)], M. Takagi [B)], S. Yoshida [B)]

[A)] High Energy Accelerator Research Organization (KEK), Tsukuba, 305-0801, Japan
[B)] Kanto Information Service (KIS), 8-21, Bunkyo, Tsuchiura, 300-0045, Japan

Abstract

A dedicated controller for the beam-position monitor (BPMC) has been developed at KEK, which is scheduled to be installed into the 50-GeV main ring of the J-PARC[1] [1, 2] project. The data acquired from a beam-position monitor (BPM) is calculated in the BPMC, and the result is sent to an EPICS device controller.

An EPICS[2] device controller for the BPMC is presently under development. It uses a VME-bus computer with an Intel processor, and runs a Linux operating system. The development started with 16 BPMCs. The performance of system-loading has been analyzed, including the usage of CPU power, the usage of memory, and the network traffic. The result shows that the EPICS device controller has sufficient capacity to control 16 BPMCs.

BPMC OVERVIEW

BPMC [3, 4] is a dedicated controller for the BPM, which was fabricated by Mitsubishi Electric Corporation. In the whole main ring, there will be 190 BPMCs with 13 groups, and each group will include a maximum 16 BPMCs.

Each BPMC has 3 modules [3], including an analog input module, an analog-to-digital (AD) conversion module, and a CPU module (see Fig. 1) [3]. Four signals from one BPM are fed to one BPMC.

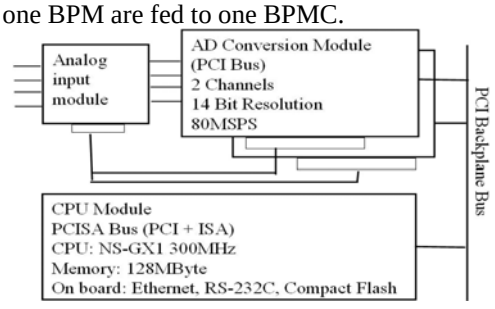

Figure 1: BPMC hardware architecture.

The BPMC picks up the signal from BPM, adjusts the signal level, digitizes it, and processes it with some specific algorithms. The result is sent to an EPICS device controller through the network. The entire procedure of signal processing for 2 channels is illustrated in Fig. 2 [5].

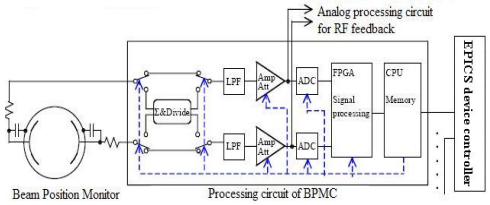

Figure 2: Entire procedure of signal processing.

Signal acquisition

The signals from a BPM are picked up by the analog input module of BPMC. In order to avoid noises from the RF subsystem and pulsed power supplies of the magnet, a differential input is used to pick up the signals from the BPM.

A Five-pole Butterworth low-pass filter with a 10 MHz cut-off frequency is adopted to provide anti-aliasing [6].

The signal level is adjusted by attenuating or amplifying the amplitude. To keep the balance between the amplification and the attenuation, a reference circuit is used.

Signal processing

The analog signal is digitized in the AD module, which includes 2 conversion channels. Each channel converts 2 input analog signals to digital signals.

The signal is sampled by 14-bit resolution with a maximum speed of 80 MSPS(s) (Mega samples per second). It supports some functions to calculate the position of each beam bunch, measure the shape of the beam signal, or observe the closed-orbit distortion during beam injection, acceleration, and quick extraction.

In addition, the data calculation is implemented in a FPGA (Field Programmable Gate Array) chip.

Processing control

The signal processing control is implemented in the CPU module. After receiving a start command from the EPICS device controller, the CPU module sets up some necessary parameters to the AD module and triggers it to start.

The calculating result, which is stored in an internal memory of the AD module, is transmitted to the CPU module through the PCI backplane bus. The CPU module then sends the result to an EPICS device controller through the network.

Fig. 3 shows the whole procedure of signal-processing control. A standard repetition cycle is 3.64 seconds, including sampling, calculating and data transmitting [3].

E-MAIL: <shengb@post.kek.jp>
[1] J-PARC: Japan Proton Accelerator Research Complex
[2] EPICS: Experimental Physics and Industrial Control System
[*] Products of the GE Fanuc Embedded System [7]

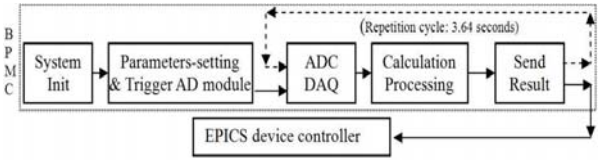

Figure 3: Signal-processing control.

EPICS DEVICE CONTROLLER

An early study of the EPICS device controller development was made with a VMIVME7805*, an Intel-based single board computer using a VME-bus (VME-SBC) [4]. It is configured as an Intel Pentium4-M processor (2.2 GHz), 1 GB memory. The operating system is booted from a local Compact Flash disk, which has 1 GB capacity.

Considering the thermal problem, a new VME-SBC (VMIVME7807*) has been introduced, which is configured with an Intel Pentium-M processor (1.8 GHz).

To reduce the system cost and improve the system maintainability, a network-booting mechanism is introduced to replace booting from a local disk. The EPICS device controller gets the Linux kernel using TFTP (Trivial File Transfer Protocol), and mounts the root directory from a NFS (Network File System) server.

The Linux kernel has been customized to support booting from the network. A related Ethernet chip driver has been compiled into the kernel. Also, some network functionalities, including NFS supporting, have been compiled into the kernel to support mapping a root directory from a NFS server.

The system configurations of 2 EPICS device controllers are listed in Table 1.

Table 1: System configuration.

Module	VMIVME7807	VMIVME7805
Processor	Pentium-M 1.8 GHz	Pentium4-M 2.2 GHz
Memory	1 GB	1 GB
Local disk	--	1 GB Compact Flash
Booting	Network	Local disk
Root	NFS ROOT	Local disk
OS	REDHAT 9	DEBIAN 3.1
Kernel	2.4.20-31.9custom	2.4.27

The hardware configuration of the development is showed in Fig. 4.

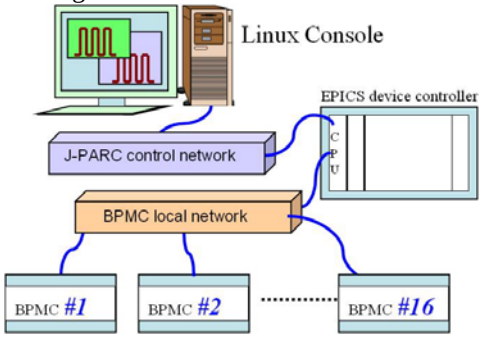

Figure 4: System hardware configuration.

This EPICS device controller supports two system services, SSHD (Secure Shell Daemon) and DHCPD (Dynamic Host Configuration Protocol). SSHD provides secure encrypted communications between the Linux console and the EPICS device controller. DHCP service is necessary to assign IP addresses of BPMCs during BPMC booting. 16 BPMCs are used for development.

2 separated networks are used to avoid the heavy network traffic caused by the BPMCs.

PERFORMANCE EVEALUATION

EPICS software

The development environment is implemented under EPICS release 3.14.6 [8]. The EPICS device controller uses a network-based device driver, NetDev [9], which provides a common framework to support intelligent network devices, including BPMC.

EPICS software is based on the concept of records, which are a set of process variables in the EPICS run-time database [8]. The property and behaviour of the EPICS device controller are predefined in the records. They are executable code units that perform hardware I/O, and provide control logic capability.

The EPICS records, which are used in the EPICS device controller for 16 BPMCs, are listed in Table 2. Totally, there are 1088 records [4]. The value from one BPMC is stored in a jpMrBpm record. The data size from one BPMC channel is 20480 bytes.

Table 2: EPICS records used in the controller.

Record Types	Total numbers
Ai	16 (1 * 16)
Ao	16 (1 * 16)
Bi	128 (8 * 16)
Bo	48 (3 * 16)
Longin	432 (27 * 16)
Longout	320 (20 * 16)
Ulongin	32 (2 * 16)
Ulongout	32 (2 * 16)
mbbiDirect	16 (1 * 16)
mbboDirect	16 (1 * 16)
Calc	16 (1 * 16)
jpMrBpm	16 (1 * 16)

System loading analysis

The control panels of BPMC for parameter setting and the waveform display were developed by Mitsubishi with dm2k, a standard EPICS GUI (Graphic User Interface) editor. Fig. 5 is a screen snapshot of a parameters-setting panel for a BPMC.

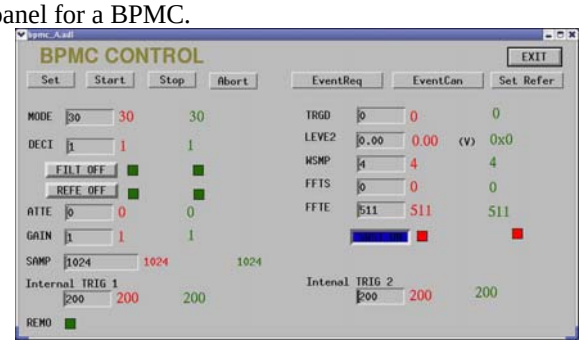

Figure 5: BPMC parameters setting panels.

Because the BPM system of J-PARC MR is under construction, to evaluate the EPICS device controller, the BPMC uses some dummy analog signals to simulate the input signals from BPM.

The CPU usage, memory usage and network traffic are analyzed. During system-loading analysis, the control panels are running on a Linux console with a 16-waveform display (see Fig. 6). Each waveform includes 200 points, and is updated every 3.64 seconds.

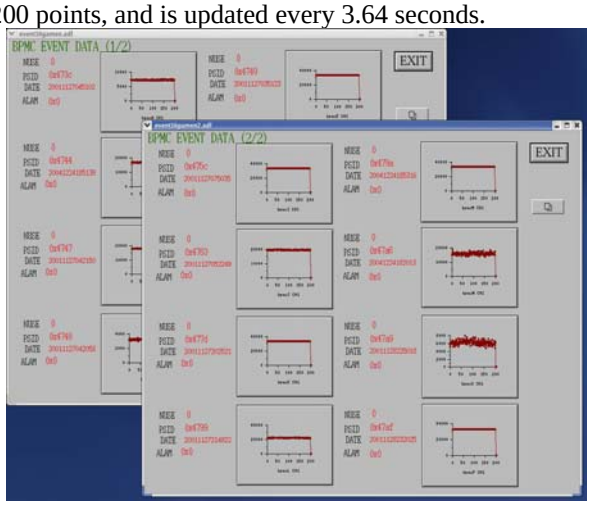

Figure 6: Waveform display panels of the BPMC.

At a run-time, the usage of CPU power is around 3.0%, and the memory usage is about 8.5% for VMIVME7805 [4]. For VMIVME7807, the usage of CPU power is around 7.0%, and the memory usage is about 11.4%.

The usages of CPU power and memory are larger in the case of VMIVME7807, because the CPU frequency of VMIVME7807 is lower, and it uses the NFS file system, even for the ROOT directory.

With the VMIVME7805 the number of network packets per second at the BPMC side is less than 650, and the average packet size is about 1066 bytes. Thus, the network traffic between the EPICS device controller and the BPMC is about 5.5 Mbps [4]. For the VMIVME7807, it is about 5.4 Mbps.

With the VMIVME7805, the number of network packets per second at the console side is about 160 and the average packet size is about 660 bytes. Thus, the network traffic between the EPICS device controller and the Linux console is about 0.84 Mbps [4]. For the VMIVME7807, it is about 0.69 Mbps.

The results are summarized in Table 3, which shows that both of the 2 EPICS device controllers have sufficient capacity (CPU, memory, and network throughput) to control one group of 16 BPMCs.

Table 3: Results of system-loading analysis.

Module	VMIVME7807	VMIVME7805
Booting method	Network	Local disk
CPU usage	~ 7.0%	~ 3.0%
MEM usage	~ 11.4%	~ 8.5%
Network traffic (BPMC side)	~ 5.4 Mbps	~5.5 Mbps
Network traffic (Console side)	~ 0.69 Mbps	~ 0.84 Mbps

SUMMARY

The BPMC is a network-based intelligent device, which is a dedicated controller for the BPM of J-PARC MR facility.

Two different EPICS device controllers for the BPMC have been evaluated. One is to use a VMIVME7805 VME-SBC, which boots the operating system from a local Compact Flash disk. Another is to use a VMIVME7807 VME-SBC, which boots the operating system from the network.

The system-loading performance for those 2 EPICS device controllers has been analyzed, including the usage of CPU power, the usage of memory, and the network traffic. The evaluation results show sufficient capacity of those 2 controllers to control 16 BPMCs.

ACKNOWLEDGEMENT

The authors would like to thank Mr. Jun-ichi Odagiri at the KEKB control group for developing the netDev and giving much helpful advice. They would also like to thank Dr. Youichi Igarashi and many others of the physics division at KEK for sharing their experiments of using Debian on VME-SBC. They would also like to thank Dr. Takashi Obina at the control group of KEK photo factory for sharing his experiments of booting Linux from the network. Finally, many thanks go to Dr. Takeshi Toyama at the beam-diagnostic group of J-PARC for his kind collaboration.

REFERENCES

[1] http://www.j-parc.jp/
[2] Y. Yamazaki, "The JAERI-KEK Joint Project for the High-Intensity Proton Accelerator, J-PARC", Proc. PAC 2003, May, 2003, Portland, p.576-580
[3] Mitsubishi Electric Corporation, BPMC manual “INS-01-0061A”, March 2003
[4] G. Shen, et. al., “DEVELOPMENT OF LINUX-BASED IOC WITH A VME-BUS COMPUTER”, Proc. PCaPAC 2005, March 2005, Hayama, Japan, KEK-Preprint 2005-14
[5] T. Toyama, “Beam Diagnostics, 50 GeV MR”, the 3rd Accelerator Technical Advisory Committee Meeting for the High-Intensity Proton Accelerator, March 2004, JAERI, Tokai, Japan
[6] T. Toyama, et. al., “Beam Diagnostics for the J-PARC Main Ring Synchrotron”, RPAT005, Proc. PAC05, May 2005, Tennessee, USA
[7]http://www.gefanuc.com/en/ProductServices/embedded/sbc_sb/vme/index.html
[8] http://www.aps.anl.gov/epics
[9] J. Odagiri, et. al., “EPICS Device/Driver Support Modules for Network-based Intelligent Controllers”, ICALEPCS’03, Gyeongju, October 2003, p.494-496