DEVELOPMENT OF LINUX-BASED IOC WITH A VME-BUS COMPUTER

Guobao Shen[A) *], Junichi Odagiri [A)], Norihiko Kamikubota [A)], Noboru Yamamoto [A)], Kazuro Furukawa [A)], Hidetoshi Nakagawa [A)], Tadahiko Katoh [A)], Susumu Yoshida [B)], Makoto Takagi [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

EPICS is a control software tool-kit used to develop control systems for accelerators and experiments. The original EPICS assumes the use of VME-bus computers, and a commercial license of a real-time operating system (VxWorks). However, recent EPICS versions (after 3.14) allow the use of Linux instead of VxWorks.

Intel-based single-board computers with a VME-bus have been available on the commercial market for long time. Thus, increasing interests have arisen to introduce Linux-based VME-bus computers with Intel CPU boards into EPICS. At KEK, the development of a Linux-based IOC has started using Linux and an Intel-based VME CPU board.

This paper describes the recent development status. It includes 2 parts. One is a system-loading analysis of an IOC, running EPICS with databases for dedicated beam-position monitor controllers. Another is a preliminary report on interrupt performance measurements.

INTRODUCTION

EPICS 3.14 and Linux-based IOC

EPICS (Experimental Physics and Industrial Control System) is a control software tool-kit for a distributed control system, which was co-developed by Los Alamos National Laboratory and Argonne National Laboratory [1, 2]. Originally, IOC (Input Output Controller) of EPICS needed (a) a commercial license of a real-time operating system (VxWorks) as an underlying target system, and (b) VME-bus computers.

From the version of EPICS 3.14, which was released in 2002, an IOC can run on many operating systems, such as Linux. This new feature becomes possible by introducing an operating system independent layer into the iocCore (the EPICS target system) [3].

Intel-based single-board computers with a VME-bus (hereafter Intel-based VME-SBC) have been available on the commercial market for a long time. In general, Intel-based VME-SBCs have a higher performance at a lower cost than PowerPC/68K-based CPU boards. Taking into account the cost for VxWorks, using an Intel-based VME-SBC for Linux-based IOC is effective to reduce the total system cost.

Actually, some accelerator laboratories have already introduced personal computers (PCs) for their control systems. The CLS (Canadian Light Source) used some industrial PCs and some Intel-based embedded PCs (PC-104) as IOCs [4]. The SNS (Spallation Neutron Source) "will also continue to deploy its control system on top of commodity-type PCs with a mix of both VME/VXI and PC front-end processors."[5]

*e-Mail: <shengb@post.kek.jp>

J-PARC and BPMC

J-PARC (Japan Proton Accelerator Research Complex) is a high-intensity proton accelerator facility. This project is a joint project between JAERI (in Tokai) and KEK (in Tsukuba), and is now under construction at the Tokai site [6]. In KEK, a dedicated controller for beam-position monitors (hereafter BPMC) has been developed by Mitsubishi Electric Corporation, which will be used in the 50-GeV main ring (MR) [7]. In the whole main ring, there will be 208 BPMCs with 13 groups (see Fig. 1).

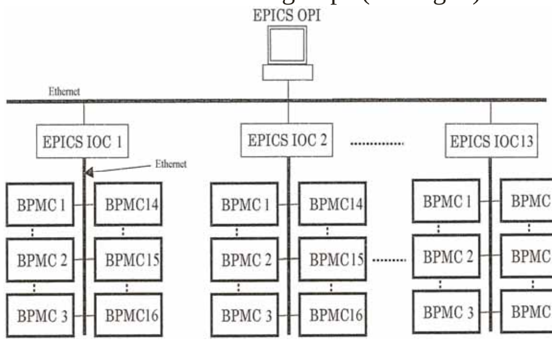

Figure 1: BPMC Configuration of J-PARC MR.

BPMC and NetDev

The IOC for BPMCs uses a network-based device driver, NetDev [8, 9]. The original NetDev was developed for commercial PLCs (Programmable Logic Controller) with a network interface, such as FA-M3 (Yokogawa), MELSEC-Q (Mitsubishi), and CVM1/CS1 (Omron). NetDev provides a common framework to support different types of intelligent devices, as illustrated in Fig. 2. Since the BPMC was designed to have a similar communication protocol as those of PLCs, adding BPMC to NetDev was easy.

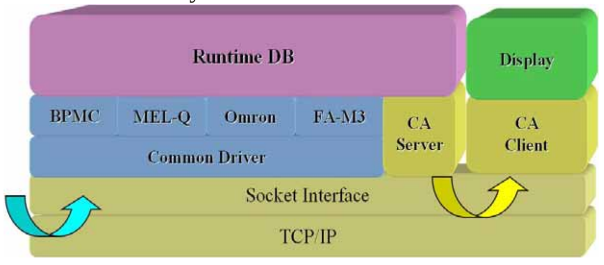

Figure 2: NetDev Communication Mechanism.

IOC SYSTEM LOADING ANALYSIS

Configuration of experimental setup

The development of a Linux-based IOC using an Intel-based VME-SBC started with 16 BPMCs. This IOC is expected to be used for one group of the BPMC controllers. The configuration of the experimental setup is illustrated in Fig. 3.

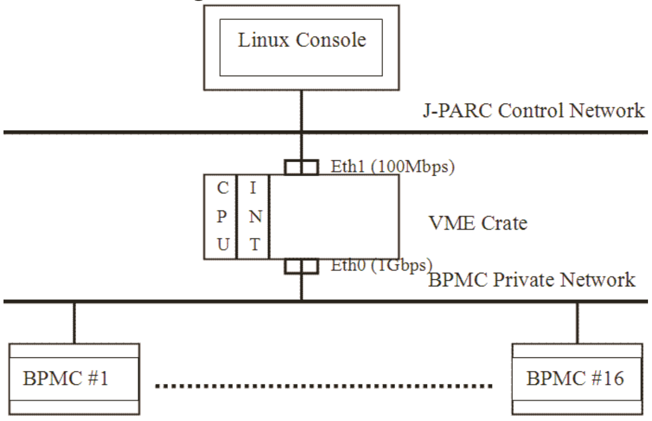

Figure 3: Experimental setup.

The CPU board is a VMIVME7805, an Intel-based VME-SBC from the VMIC Company, GE Fanuc Embedded System [10]. The CPU is configured with a Petium4-M (2.2 GHz), 1 GB memory, a Compact Flash disk of 1 GB capacity, and 2 Ethernet ports. The Eth0 (1Gbps) is connected to the BPMC private network, and the Eth1 (100Mbps) is connected to the J-PARC control network. A standard PC (Linux Console) is connected to the J-PARC control network. GUI-based control panels for the BPMCs run on it (Fig. 4). There is an “INT” board in the VME crate, which is used to generate a VME-bus interrupt to evaluate of interrupt performances.

The development environment is implemented under EPICS release 3.14.6 [11]. One of popular Linux distributions, Debian 3.1 (Sarge) [12], is used as the operating system with the EXT2FS file system, which uses Linux kernel version 2.4.27.

The VMIVME7805 uses a PCI-VME interface chip, Tundra Universe II, to access the VME bus. The VME-bus driver for Linux, vme_universe, is a part of the BSP (Board Support Package), which is available for free under the BSD license. The BSP version that we have used is vmisft-7433-3.3.

On the IOC, two system services, SSHD and DHCPD, are activated. SSHD (Secure Shell Daemon) provides secure encrypted communications between the console and the IOC. Most of the developments have been made at the Linux console using the SSH protocol. DHCP (Dynamic Host Configuration Protocol) service is necessary to assign IP addresses of BPMCs during BPMC booting.

IOC Loading Analysis

We have implemented a stand-alone EPICS system on a Compact Flash disk. The disk includes Debian basic system, development tools, a Universe II chip driver, and all of the EPICS base (source, object and binary) files. EPICS databases for the 16 BPMCs are also included. The total disk usage is 370 MB.

All of the database records for the 16 BPMCs are listed in Table 1. Totally, there are 1088 records. The jpMrBpm record is used to store waveform data of the BPMC. Each BPMC has 4 channels, and the data of each channel is 20480 bytes. The signal of each channel is sampled with a speed up to maximum 80MSPS (Mega samples per sec) at 14 bits. After calculating the sampling data by a specific software algorithm, the value of each BPMC (proportional to the signal amplitude) is transmitted to IOC. The time of a standard repetition cycle is 3.64 seconds, including sampling, calculating and transmitting data to IOC.

Table 1: Records used in the IOC

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)

The control panels of BPMC were developed by Mitsubishi with a standard GUI editor, dm2k. When the control panels are running with a 16-waveform display (Fig. 4), the number of network packets per second at the console side is about 160, and the average packet size is about 660 byte. Thus, the network traffic between the IOC and the console is about 0.85Mbps.

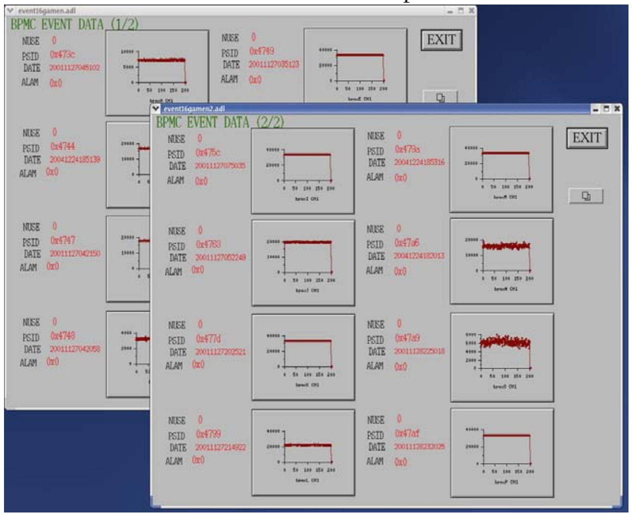

Figure 4: BPMC control panels.

At a run-time, the usage of CPU power is 1~3% and the memory usage is about 8.5%. 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 IOC and the BPMCs is less then 6 Mbps (5.5 Mbps in average). Since the Ethernet port is 1 Gbps, the network traffic corresponds to less than 1%. The results are given in Table 2, which show that the VMIVME7805 has sufficient capacity (CPU, memory, and network throughput) to run one group of BPMCs.

Table 2: Results of IOC Loading Analysis.

CPU	MEM	Network (BPMC side)	Network (Console side)	Disk Usage
1~3 %	~8.5%	~ 6 Mbps	~ 0.85 Mbps	370 MB

INTERRUPT PERFORMANCES

EPICS VME Library and Interrupt Handling

The EPICS VME library, vmeUniverse, is being developed at KSTAR [13]. The library is designed to support a set of functions similar to VxWorks, in order to enable easier porting of the existing hardware drivers of VxWorks to Linux. The EPICS VME library (vmeUniverse) uses functions of the VME-bus driver of BSP (vme_universe).

More work towards full-featured support is in progress. The existing version of the EPICS VME library supports the following functions:

- Map VME address space into a processor's address space, vmeUniverse_sysBusToLocalAddrs().
- Connect interrupts with user-specified routines, vmeUniverse_intConnect().
- Generates a local-bus interrupt, vmeUniverse_generateVMEInterrupt().

The interrupts can be generated by two different methods.

- (1) One is to call a special function of the VME-bus driver, which triggers a register of the Universe II chip and generates an interrupt.
- (2) Another is to uses a dedicated VME board, "INT" in Fig. 3, which generates a VME-bus interrupt when a write action is made into a special register.

During a measurement with a dedicated VME board, sometimes a user-specified interrupt routine gets called twice upon a single interrupt. This is because a re-enabling interrupt is made at an inappropriate position in the original VME-bus driver. After moving the function of enabling an interrupt to the EPICS VME library, everything works well [14].

Performance Measurement

The interrupt response times are measured by two interrupt methods. The algorithm of the measurement is:

1. Read the count of the Time Stamp Counter, a special counter of Pentium CPU.
2. Generate an interrupt, by calling a function to trigger the Universe II chip, or by writing a value to the register of a dedicated VME board.
3. Read the count of the Time Stamp Counter again at the entrance of the user-specified interrupt routine.
4. Calculate the response times, taking into account the CPU clock rate.
5. Repeat the procedure to accumulate data. Actually, this is carried out by a sub thread, which is created by a shell script (usually st.cmd).

In addition, the response times are measured in 2 cases:

- Case 1: running iocCore only.
- Case 2: running iocCore with the BPMC databases.

The purpose of Case 2 is to respond to Ethernet interrupts generated by the BPMC, which make the system work heavier during the measurements.

Fig. 5 is a typical result of one measurement, which corresponds to 1,000,000 loops. It shows a histogram of the response times in Case 2, caused by using the Universe II chip. Here, we define the overhead as the minimum response time. The overhead in Fig. 5 is less than 5micro-seconds. The 90.0% of the response times are around 15 to 18 micro-seconds (3 micro-seconds width). The latency is defined as the maximum response time. The latency in Fig. 5 is less than 1386 micro-seconds.

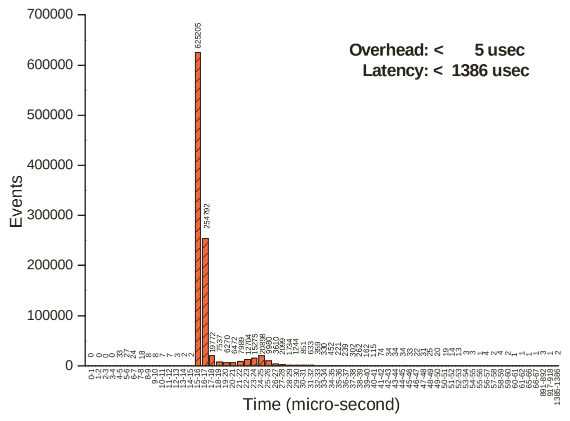

Figure 5: Interrupt response times by the Universe II chip.

There is another obvious peak at around 24 to 25 micro-seconds in Fig. 5. It may be caused by some background interrupts in the kernel space, for example Ethernet interrupts generated by BPMCs, or some clock interrupts. The user-specified interrupt routines are handled in the Linux user space. Thus, they would be suspended by kernel space interrupts.

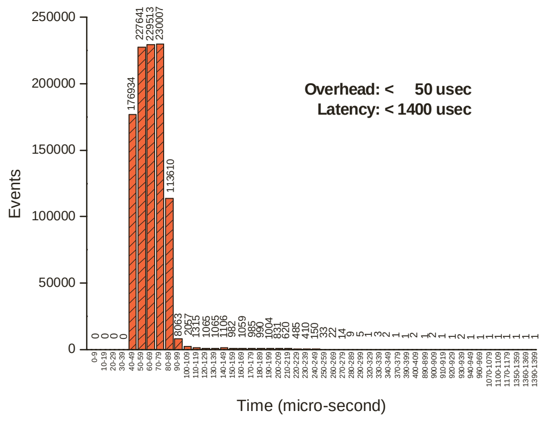

Figure 6: Interrupt response times by a dedicated VME board.

Fig. 6 shows another histogram of similar conditions to Fig. 5, but the interrupt source is a dedicated VME board. The overhead is less than 50 micro-seconds, and the latency is less than 1400 micro-seconds. About 97.8% of the response times are in the 40-90 micro-second window.

Detailed results of various measurements are given in [14]. The results of interrupt overheads and latencies are summarized in Table 3.

Table 3: Summary of the overheads and latencies in various measurements (UNIT: microsecond).

	Overhead		**Latency**	
iocCore	Case 1	Case 2	Case 1	Case 2
(1) Interrupt by Universe II Chip	< 5	< 5	<1341	<1386
(2) Interrupt by Dedicated board	< 50	< 50	< 890	< 1400

A comparison between Case 1 and Case 2 gives the following results: the overheads show the same time values for 2 different cases, but the latencies show worse values in Case 2 when interrupts are caused by a dedicated VME board. For different interrupt methods, the latencies show the same value as in Case 2, but the overheads show worse values when interrupts are caused by a dedicated VME board.

Discussion

It should be mentioned that the interrupt latencies are rather preliminary, because it is apparent that more loops will result in longer latencies. Moreover, interrupts are triggered by software in both cases. As a result, interrupts occur synchronously all the time, which does not simulate a real world.

The typical response times are different between two interrupt methods (15~18 micro-seconds vs. 40~90 micro-seconds, by a Universe II chip vs. by a dedicated VME board).

In Fig. 5, a small number exist in the 4~5 micro-second range, while most of the response times are in the 15~18 micro-second range.

No study was made to understand time differences.

CONCLUSION

When the BPMC iocCore runs on a VMIVME7805, the usage of CPU power is less than 3%, and memory usage is about 8.5%. The network traffic is less than 6 Mbps (about 5.5 Mbps) between the IOC and the BPMC, and less than 1 Mbps (about 0.85 Mbps) between the IOC and the OPI. This result shows sufficient capacity of VMIVME7805 to run one group of BPMCs (16 BPMCs).

The present measurements show that the latency is less than 1400 micro-seconds, and that the overhead is less than 50 micro-seconds, under various conditions. In the future, more rigorous measurements should be made, especially for the latency evaluation.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Kukhee Kim at KSTAR for providing the EPICS VME library. They would also like to thank Dr. Youichi Igarashi and many others of physics division at KEK for sharing their experiments of using Debian on VME-SBC. Finally many thanks go to Dr. Marty R. Kraimer at Argonne National Laboratory for his helpful advices.

REFERENCES

[1] http://www.aps.anl.gov/epics/index.php

[2] R. Lange, J. B. Anderson, A. N. Johnson, M. R. Kraimer, W. E. Norum, L. R. Dalesio, J. O. Hill, “EPICS: Recent Developments and Future Perspectives”, Proc. of Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS’03), Gyeongju, October 2003, p.278-281.

[3] M.R. Kraimer, J.B. Anderson, J.O. Hill, W.E. Norum, “EPICS: A Retrospective on Porting iocCore to Multiple Operating Systems”, ICALEPCS’01, San Jose, November 2001, p.238-240.

[4] http://www.lightsource.ca/machine/pdf/7.4.39.1.rev.2-control_system_technical_specification-matias.pdf

[5] E. L. Williams Jr., G. S. Lawson, “CONTROL SYSTEMS ON LOW COST COMPUTERS”, Proc. PAC 2003, May, 2003, Portland, p.288-290

[6] Y. Yamazaki, "The JAERI-KEK Joint Project for the High-Intensity Proton Accelerator, J-PARC", Proc. PAC 2003, May, 2003, Portland, p.576-580

[7] H. Nakagawa and T. Toyama, KEK, private communication

[8] J. Odagiri, J. Chiba, K. Furukawa, N. Kamikubota, T. Katoh, H. Nakagawa, N. Yamamoto, M. Komiyama, I. Yokoyama, H. Song, Y. Yamamoto, H. Miyaji, H. Satoh, M. Sugimoto, “EPICS Device/Driver Support Modules for Network-based Intelligent Controllers”, ICALEPCS’03, Gyeongju, October 2003, p.494-496

[9] K. Furukawa, J. Chiba, N. Kamikubota, H. Nakagawa, “Implementation of the EPICS Device Support for Network-based Controllers”, ICALEPCS’01, San Jose, November 2001, p.197-199

[10] http://www.vmic.com

[11] http://www.aps.anl.gov/epics/base/R3-14/6.php

[12] http://www.debian.org

[13] K. Kim, KSTAR, KSBI, private communication

[14] G. Shen, “Linux-based IOC by using Intel-based VME-SBC”, to be submitted to KEK-Internal Report