AN EMBEDDED LINUX SYSTEM BASED ON POWERPC

Ye Mei, Guanjuan, Tian Yurun, IHEP, Beijing, P.R. CNINA
Chu Yuanping, Zhu Kejun, Zhao Jingwei, IHEP, Beijing, P.R. CNINA

Abstract

This article introduces an Embedded Linux System based on VME series PowerPC as well as the essential method on how to set up the system. The goal of the system is to build a testbed for VMEbus device of Front-End Electronics. It also can be used to set up the data acquisition and control system. Two types of compiler are provided by the developer system according to the features of the system and the PowerPC. At the top of the article some typical embedded Operation system will be introduced and the features of different system will be provided. And then the method on how to build an embedded Linux system as well as the key technique will be discussed in detail. Finally a successful data acquisition example will be given based on the test system.

INTRODUCTION

It has been 12 years since the commission of the Beijing Spectrometer (BES) at Beijing Electron Positron Collider (BEPC). Both machine and detector have undergone the upgrade. According to the machine design and physics goals, BEPCII is designed with a peak luminosity of $10^{33}cm^{-2}sec^{-1}$. After Level 1 trigger, the estimated event rate will be about 4000Hz at J/ Ψ peak. The total channel number of electronics will be up to 40K, among which there will be 30K ADCs and TDCs. The new electronics will up to higher event rate and has much more noise environment. The DAQ system will adopt embedded real time operation system (RTOS) and corresponding programming techniques to fit the readout requirement of the Front-End Electronics (FEE).

Considering the high price of the commercial RTOS such as VxWorks, LynxOS, the main purpose of this paper is to discuss the possibility of using the embedded real-time Linux as the operating system of the readout system.

The Readout System

The main tasks of the readout system are collecting event data from the front-end electronics after Level 1 trigger; transferring data fragments from the VME readout crate to the readout PC which related to online computer farm through two levels of computer pre-processing and high speed network transmission. The system is designed to adopt Motorola's PowerPC as the front-end processor to readout data from the FEEs. On the basis of real-time operation system, the advanced computer and network technologies, multi-level parallel data collecting/processing schemes will be adopted. The readout system is intended for work with the readout electronics. Before the start of run, it performs initialization and calibrations of sub-systems; during the run, it reads data from the VME readout crates, each holding a system controller and some FEE readout modules (ADC and/or TDC). The VME processor, PowerPC, an embedded single board computer is used to collect, pre-process and transfer data. Several readout crates are connected to a readout PC through fast Ethernet, thus constituting a readout branch. All the readout branches are connected to the online computer farm through gigabit switch, constituting the backbone of the DAQ data flow. Sub-event packages, coming from each readout branch, are assembled in the online computer farm. After being filtered and processed, the events are recorded in persistent media.

RTOS

As a general-purpose operating system, Linux is optimized for average performance across all types of applications. While this is suitable for most desktop and server environments, it poses problems for environments where the performance or behavior of some applications is sensitive to the latency of response. The TimeSys Linux kernel enhancements are geared to improve and give better control to application developers on the latency of response that they can achieve. These enhancements include a fully preemptible kernel, schedulable hard and soft interrupt processing, an enhanced low-latency scheduler, and improved process time accounting.

TESTBED ARCHITECTURE

In order to check if the designed system can read out large amount of data from the FEEs and fit the requirement of high frequency data acquisition, The testbed is designed on a VME crate as figure 1 showed. A PowerPC named MVME5500 is used as the VME processor to collect data from FEEs. Another PowerPC named MVME2431 acts as electronic modules, which can generate VME interrupt and Monte Carlo (MC) data. Both of the PowerPCs are placed at the same VME crate. An x86 PC so called host machine supports develop console, net file system (nfs) and related cross compiling tools, on which installed with TimeSys Software Development Kit (SDK) and TimeSys Storm, an Integrated Development Environment (IDE). The PC installed TimeTrace, a Windows-based profiling environment, could provide execution data from the runtime target and reveals the critical events happening in the application, such as context switches, timer events, scheduling events, and more, so the problem areas can be pinpointed through the Ethernet. The data fragment collected by the MVME5500 will be packed and

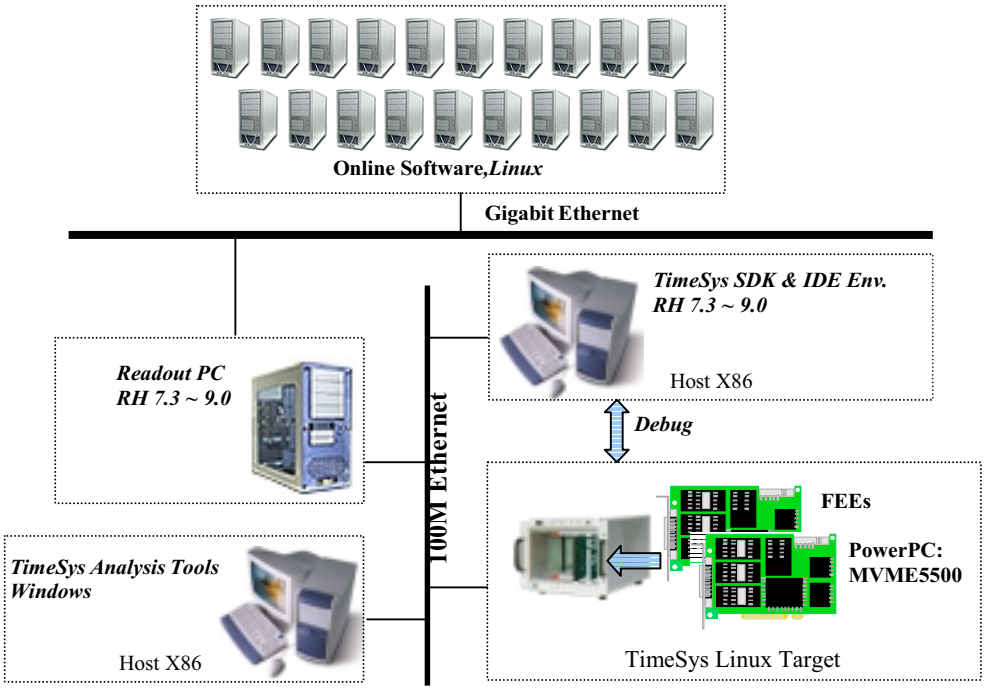

Figure 1: The system Architecture of the experimental environment

transferred to the readout PC, which pack the data of the several crates and submit to the online software. To accomplish above tasks, the experiment system must be designed to have the following functions:

- To collect data from the VME readout modules (ADC and TDC) at high speed and full utilize the VME bus bandwidth;
- To set up the multi-level buffering techniques that solve the "bottleneck" problem caused by the VME data fragments assembling and the network transmission;
- To use the necessary task synchronization strategy and software protocols that ensure the correct parallel processing;
- To build the effective channel between the kernel module and the user module;
- To provide the corresponding program that simulate the Front-End Electronics modules to generate the VMEbus trigger and Monte Carlo data;
- To establish the APIs between the readout code and the online software.

The PowerPC Board: MVME5500

The MVME5500 showed in figure2 is a single-board computer based on the PowerPC MPC7455 processor and the Marvell GT-64260B host bridge with a dual PCI interface and memory controller. On-board payload includes two PMC slots, two SDRAM banks, an expansion connector for two additional banks of SDRAM, 8MB boot Flash ROM, one 10/100/1000 Ethernet port, one 10/100 Ethernet port, 32MB expansion Flash ROM, two serial ports, NVRAM and a real-time clock. The MVME5500 interfaces to a VMEbus system via its P1 and P2 connectors. The 1M Flash ROM was configured a

Figure 2: PowerPC, MVME5500

firmware package named MOTLoad. The main function of the MOTLoad firmware package is to serve as a board power-up and initialization package, and to serve as a vehicle from which user applications can be booted.

DESIGN AND IMPLEMENTATION

Develop Environment and the Linux Kernel Configuration

The TimeSys Linux distribution provides a complete Linux runtime and development environment that executes on the target board and reference host machine including compilers, debuggers, and other utilities as command-line. The TimeSys Linux SDK includes the GNU C language compiler. One of the most interesting aspects of GNU C compiler is that it is designed for use as both a native and a cross compiler.

- The cross compiler, which executes on the desktop host machine development system but produces

binaries that can run directly on the target hardware: PowerPC.

- The native compiler for the target board, which executes directly on the PowerPC and produces binaries for it. The gcc is located in the NFS-mounted root file system used by the SDK when booting Linux on the target board.

The TimeStorm is an integrated development environment for Linux that runs on both Linux and Windows systems. It provides a graphical display of local and remote debugging. It is also tightly integrated with the tools provided in the SDK.

The runtime kernel was precompiled with an export NFS root file system. However, in some cases, we should have to recompile the kernel in cases of adding or removing device drivers, protocols, or simply to more accurately reflect particular hardware configuration. A kernel configure file named '.config' is provided. The UNIVERSE driver, IP setting and DHCP configuration of the PowerPC was changed and rebuild through the configure file.

BSP: The Universe II

The Universe II (CA91C142) is the de facto industry standard PCI bus to VMEbus bridge, providing 64-bit, 33 MHz PCI bus interface, produced by Tundra Co. The function involved in interfacing VMEbus to PCI includes: address mapping, byte-lane swapping, and cycle mapping. Which can provide multi-master, multi-processor architectures on VMEbus systems using PCI.

System Latency

In order to measures the ability of the system to meet latency or response time requirements we test on the following ways of measuring RTOS predictability and performance:

- VME read and write(R&W) time
- Task Context Switch Time
- Network overhead
- DMA read overhead
- VME interrupt response time

The VME R&W time measures the system latency on behave of VMEBus read and write operation, the results are 1942 ns and 407 ns respectively.

The task context switch time assesses the most common types of overheads in a system. These are measured in the absence of resource contention, and provide information on the performance of the operating system for throughput sensitive applications. To measure the context switch time, two threads with identical real-time priorities are set up. Both threads execute a while loop, and continually yield to the other thread by calling the sched_yield() system call. Timestamps are taken before and after yield. The timestamps from the two tasks are post-analyzed to generate context switch times. Context switch time is measured under a "no-load" scenario: no additional non-system processes are created to increase the system load. The context switch time as table 1 showed measures the observed value of context switch time, and at a minimum includes the raw context switch time, the scheduling overhead (which is minimal for the way the test is set up), the cost of sched_yield (), and the cost of crossing the user-kernel boundaries. In addition, in some cases the observed value may include the cost of handling any interrupts, and any blocking from shared locks in the kernel.

Table 1: Context Switch Time Summary
(Numbers given in Nanoseconds)

Context Switch			
Mean	**Min**	**Max**	**Std Dev**
3000	3000	3000	0

About the networking performance, the system occupies about 15% of CPU time for sending 1024 Bytes data block continually with about 93Mbps throughput.

The DMA read latency time measures the interval spending on the whole DMA read operation including start the DMA read, complete the data transfer and receive the DMA controller interrupt return. We test the value from 1Byte through 16,384Bytes, which has the graph as below:

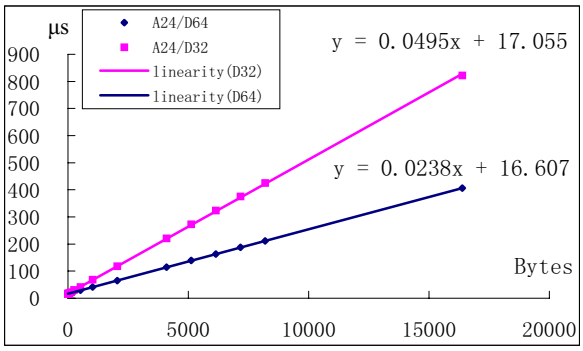

Figure 3: DMA Overhead

From above figure we can see two lines. One is the trend line of reading A24/D32 VME data block and the other is reading A24/D64 VME data block. Both have the system overhead of 17 μ s through the DMA transfer. This overhead includes DMA initialisation, DMA controller starting, the latency of DMA interrupt response and the time returning to the user level.

The VME interrupt response time is the length of time between when the VMEbus hardware interrupt is asserted and the time corresponding task begins execution. This includes the interrupt latency time, scheduling time, VMEbus handshake and time to return to user-level task. The handshake means we use a VME R&W to control the source interrupt device to either generate a VMEBus interrupt signal or pending. The VME interrupt response time measures the ability of the system to meet latency or response time requirements in a system. The periodic task is used to measure the interrupt response time as follows: a periodic task suspends itself to be awakened when its

period expires, and a timer that expires periodically is associated with the periodic task. A timestamp is recorded when the periodic task wakes up, and the program then records the difference between the timestamp and the period expiry time. The kernel RTOS is able to get very tight response time (average 14 μ s) under heavy network throughput and computing load. This is benefit from the changes made in the Linux kernel and real-time modules, such as adding kernel preemptibility, prioritizing IRQ and soft-IRQ activity, minimizing interrupt mask times, priority inheritance mutexes, and adding high accuracy clocks and timers etc.

The Readout Module

In the readout module as the figure 4 showed the PowerPC is used to read data from the electronic modules through VMEbus. The FEE generates VMEbus interrupt and Monte Carlo(MC) data. The MC data block size is fluctuant and has the average size of 640 bytes. To simulate the readout process three threads with different real-time priorities are set up: the dma task, the pack task and the net task. The dma task is used to response the event trigger and collects data from the electronic readout modules. The pack task builds the raw data and writes the data into another ring buffer shared by net task prepared for the network transfer. The net task sends the data to the readout PC, which will build the whole data from the subsystem and communicate with online software. Two ring-buffers are used to buffer the data: the dmaRng and the netRng. The dmaRng is set to load the raw data collected by the dma module. The netRng is set to store the built data from the pack module. Two message queues are used to ensure the threads synchronization. The reason we use message queue is the fluctuation of data block size. The message queues and the ring-buffers make the readout modules parallel as following way: Once the dma module catch the event of VME interrupt it will start transfer data and send a message to pack module telling a block of data size. And then inform the VMEbus to enable the next interrupt by a VME R&W handshake. During the DMA data transfer the CPU is released to do data packing and network transmission. The CPU overhead and the event rate are used to measure if it can fit the readout requirements.

The message queues here we used is the POSIX message queues implemented by TimeSys Linux/Real-Time in addition to the standard System V messages queues that have long been part of the Linux kernel. Traditional System V message queues have two main shortcomings in the context of real-time application, shortcomings that POSIX message queues avoid. First, the corresponding function for reading a message queue from a POSIX message queue, mq_receive (), reads the oldest message with the highest priority, while the message queue of System V can return a message of any desired priority. Second, the POSIX queues can generate a signal of start a thread when a message is enqueued on a previously empty queue. System V message queues lack such a facility.

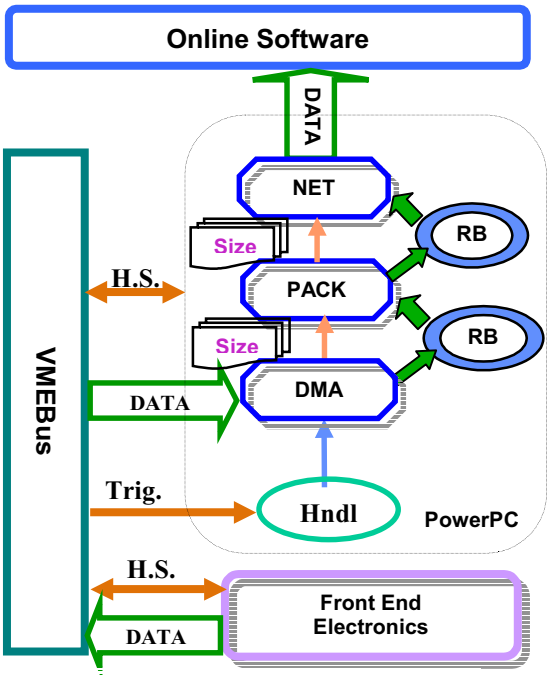

Figure 4: The Software Structure

The Response Jitter

The response jitter is the length of the response time between the two VMEbus interrupt under the heavy task context switch and network load. Measurements are conducted under the same conditions as the readout program except for the configuration of the buffer full and pack task. So that the jitter cannot affect by the task strategy controlled by the semaphore.

As the results indicate that TimeSys Linux/Real-Time is able to get very tight response time even under heavy background system load. This is a direct consequence of the changes made in the TimeSys Linux kernel and real-time modules, including adding kernel preemptibility, prioritising IRQ and soft-IRQ activity, minimizing interrupt mask times, priority inheritance mutexes, and adding high accuracy clocks and timers.

CONCLUSIONS

As experiment discussed above the strategy using the embedded real-time Linux as the operating system of the PowerPC is able to fit the essential requirement of the readout system. In addition to the task of readout and net transmission there is still enough CPU idle to ensure the stabilization of the system. Considering the nice price and performance of the TimeSys Linux/real-time the DAQ could take into account the possibility of adopting the real-time Linux.

REFERENCES

[1] http://www.timesys.com/.
[2] TimeSys Linux/Real Time User's Guide.
[3] TimeSys Benchmark Report SDK.