EMBEDDED LINUX SYSTEM FOR ACCELERATOR CONTROL APPLICATIONS

Tasaddaq A. Khan*, *NSRRC, Hsinchu, Taiwan*
Farhanullah, *Informatics Complex, Islamabad, Pakistan*
* *also informatics complex, Islamabad, Pakistan*

ABSTRACT

Accelerators have played an important role in basic and applied research. Control systems are the backbone of the accelerator facilities. The Linux operating system has a very bright future in the area of embedded applications for anything from Internet appliances to dedicated control systems. The power, reliability, flexibility, scalability of Linux, and communications protocols have established Linux as an increasingly popular software platform for a vast array of projects and products. Embedded Linux systems are cheaper solutions for various applications in accelerator control system. The architecture of the embedded Linux system used for the accelerator control application will be presented in this report.

INTRODUCTION

An embedded system is a special purpose computer system, which is completely encapsulated by the device it controls. These systems are usually designed to perform selected functions at low cost and reside in machines that are expected to run continuously for years without errors. There are many different CPU architectures used in embedded designs such as ARM, MIPS, PIC, 8051, and Power PC etc. A common configuration for very-high-volume embedded systems is the system on a chip (SOC), an application-specific integrated circuit (ASIC), for which the CPU was purchased as intellectual property to add to the IC' s design. A related common scheme is to use a field programmable gate array (FPGA), and program it with all the logic, including the CPU. Embedded system designers use compilers, assemblers, and debuggers to develop embedded system software. Embedded systems often have no operating system or a specialized embedded operating system. Embedded systems typically have a small hardware footprint, need to minimize use of computing and power sources, and operate in environments that do not tolerate rotating storage media.

FEATURES OF EMBEDDED LINUX SYSTEM

Linux is most widely used as an operating system to the embedded systems. Linux is a clone of the Unix operating system written by Linus Tovalds, a student of Helsinki in Finland, with assistance from programmers across the Internet [1]. Linux is a very capable operating system that has one very big advantage over almost all other operating systems, which is its low cost. The free nature of the Linux source code and its availability on a wide range of processor architectures has made it popular in the embedded systems community. It is very modular in nature for using in embedded systems because all features of the system that are not needed for a specific embedded system can be removed from the kernel [2]. In addition, Linux has been ported successfully to a large number of processor architectures, which allows it to run on many different types of CPUs. Linux has all the features one would expect in a modern Unix machine, including fully functioned multitasking, virtual memory, shared libraries, demand loading, proper memory management and TCP/IP networking. Many Linux flavors cater to the embedded/realtime market. These include RTLinux (Real Time Linux), uClinux (Linux for MMUless devices), ARM-Linux (Linux on ARM), Montavista Linux (Linux distributions for ARM, MIPS, PPC).

Embedded Linux development broadly involves three tiers namely the bootloader, the Linux Kernal, the graphical user interface (GUI) [3]. The boatloader is usually the first piece of code that will be executed on any hardware and initializes the system peripherals. It loads and execute Linux kernel after initializing the system peripherals. A minimal embedded linux system needs just three elements: a boot utility, the linux micro-kernel, composed of memory management, process management and timing services, an initialization process. To do anything useful while remaining minimal, one can also add: drivers for the hardware, one or more application processes to provide the needed functionality. As additional requirements become necessary, one might also want: a file system (perhaps in ROM or

RAM), TCP/IP network stack, a disk for storing semi-transient data and swap capability, a 32-bit internet CPU (required by all complete Linux systems) [4].

The one disadvantage to running Linux on an embedded system is that the Linux architecture provides real-time performance through the addition of real-time software modules that run in the kernel space, the position of the operating system that implements the scheduling by crashing the operating system, which can be a very serious vulnerability for the real-time applications [5].

GENERAL ARCHITECTURE OF ACCELERATOR CONTROL SYSTEM

Modern control system is structured into hardware and software layers that manipulate data on different levels of abstraction. The bottom layer interacts with the electrical signal where the processors have to implement real-time control. The top layer provides the human interface where operators can control the accelerator [6]. The layers in between maintain the machine parameter database, and provide data-collector, data distribution, networking, and monitoring, and control signal timing. Control system of synchrotron light sources is distributed. Whenever the processor power or data storage capability of the central component are not sufficient, the control system can be spread over several computers. This may involve distribution of software applications or splitting the machine parameter database into a distributed database. A distributed control system consists of a hierarchy of processor layer.

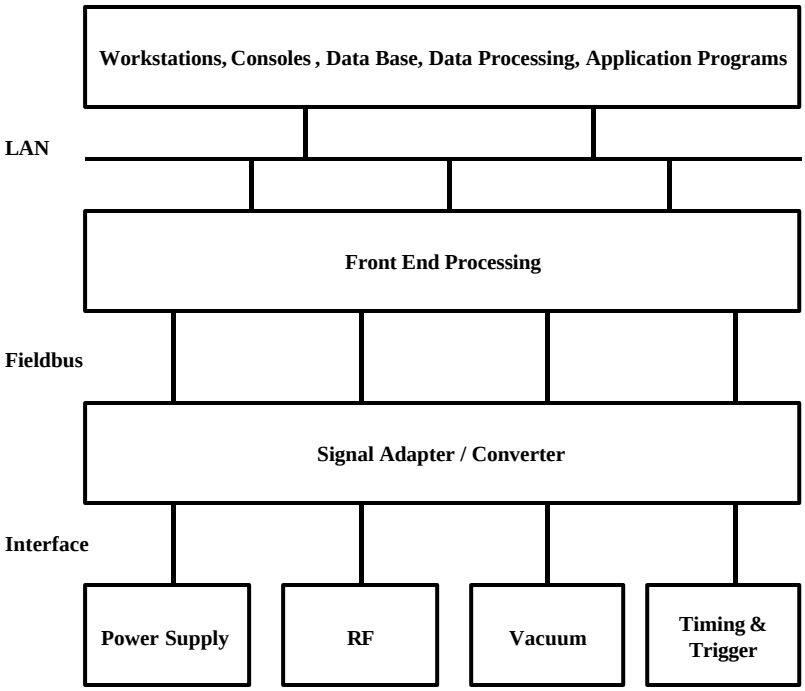

Figure 1: Schematic of control system hierarchy for hardware components

The front-end system can be characterized as the low level part of the control system. The front-end system interfaces, or connects, physical signals derived from electrical hardware components to the embedded systems or computers. VME systems are now widely used for front-end purposes in accelerator control systems [7]. The components of the accelerator control systems are linked by a network to transmit data. The use of standardized networks within a control system directly couples the control system with the computer network within the whole laboratory. Ethernet, the first high-speed local area network standard (100 Mbits/sec) is widely used at this time. The control system has to perform a series of tasks to keep the accelerator operational. Each of these tasks is controlled by software packages, commonly referred to as application programs. The most visible part of the control system is the operator interface i.e., the consoles and displays. Most systems use workstations now, but control systems based on personal computers can also be found. Monitoring of the accelerator status is

very important both at high level and front-end system. Alarms and operator actions can be logged and displayed on consoles.

PROTOTYPE CARD AS EMBEDDED LINUX SYSTEM

Due to the popularity of the embedded systems and distributed control systems, many companies have started developing low cost embedded system solutions. ARM is the industry's leading provider of 32-bit embedded RISC microprocessors with almost 75% of the market. ARM offers a wide range of processor cores based on a common architecture that deliver high performance together with low power consumption and system cost.

The ARM processor range provides solutions for [8]:

- Open platforms running complex operating systems for wireless, consumer and imaging applications.
- Embedded real-time systems for mass storage, automotive, industrial and networking applications.
- Secure applications including smart cards and SIMs.

A prototype card (NET-Start!™) having network processor S3C4510™ based on 32-bit ARM7TDMI™ core is high performance and low cost solution for network applications. The central processor core is low power 32-bit RISC macro-cell incorporating Thumb™ 16-bit compressed instruction set.

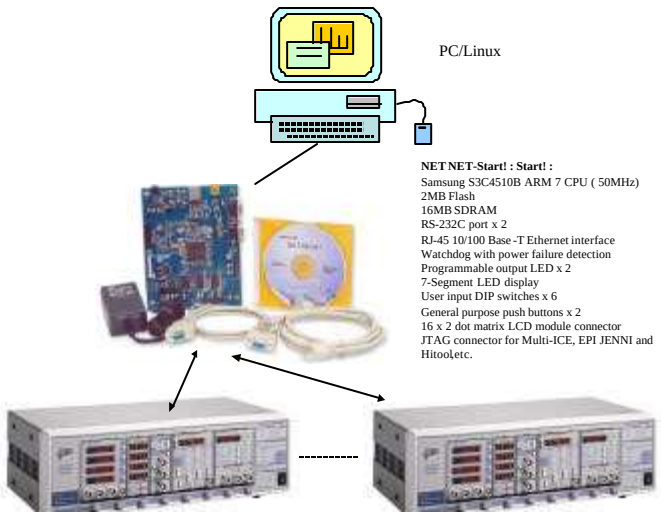

Figure 2: Prototype development environment.

In addition, the S3C4510 processor features a configurable unified 8-Kbyte cache/SRAM, an I^2C serial interface, two UARTs, two timers, 18 programmable I/O ports, and a 10/100BaseT Ethernet controller. These features make it a cheap network solution for Embedded Linux. One of the major advantages of this prototype card is that it contains all the hardware and software development tools needed to build network applications. It also contains schematics, GNU development tools, uClinux kernel source codes, and the source code of the application program. By integrating the stability and open source advantage of Linux with this cost effective and high performance microprocessor, one can start network applications development faster. Linux based C is used to develop program to measure some data and control the machine.

We tested the embedded Linux system by interfacing with SRS SIM module [9]. SIM (small instrumentation modules) is a robust, flexible platform in which up to eight high performance instruments share the same compact mainframe and computer interface. The SIM900 mainframe is the platform on which a SIM system is assembled. The mainframe provides power, computer interfaces, clock synchronization, and individual module status. Eight internal module slots accommodate single-

width and double-width modules. Mainframe comes with a standard RS-232 host interface and GPIB (IEEE-488.2) as an option. Mainframe power supply provides stable, regulated, and filtered DC voltages of ±5, ±15, and +24V to different modules.

POSSIBLE APPLICATIONS OF EMBEDDED LINUX SYSTEM IN ACCELERATOR CONTROL

Embedded Linux Systems due to their flexibility and cheap solutions are being used in many fields. It can be used to measure the beam current and calculate lifetime, measure the vacuum pressure, control of the magnet power supply, etc. in the accelerator applications.

The current in the storage ring is directly given to SIM900 module that was connected to the prototype card by RS232 and the card was connected to the computer. Figure 3 (a) shows the measured current vs. time and figure 3(b) shows the calculated lifetime. Regression method was used in the following formula to calculate the lifetime of the beam.

$$\tau_{\text{beam}} = \frac{I(t)}{dI/dt}$$

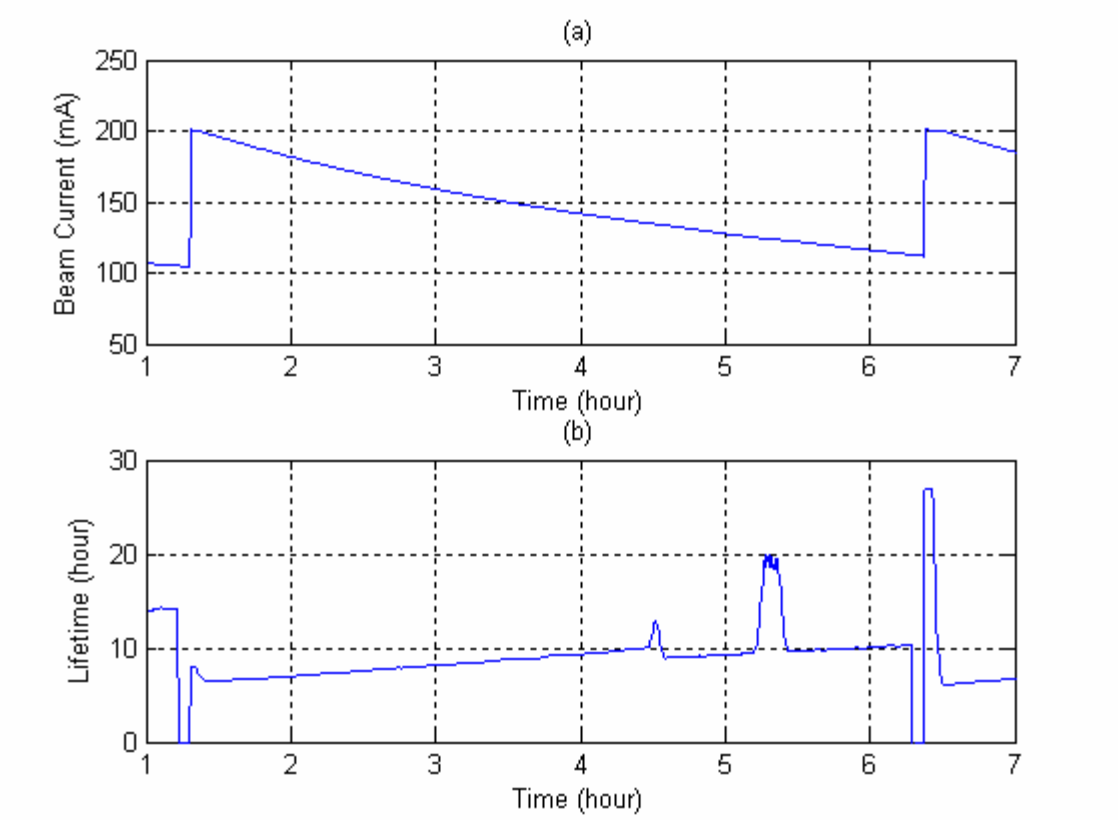

Figure 3(a): Beam Current vs. Time
Figure 3(b): Lifetime vs. Time

DISCUSSION

Embedded Linux Systems and their importance in the control system have been addressed in this report. Portable Linux is being used most widely in the control field. Many vendors are designing control cards based on Linux. These systems are being used to measure and control certain parameters in the accelerator. An embedded Linux prototype card was used to measure the different parameters of the accelerator and control them. These results were compared with the results taken by some other data bus solutions to check the performance of this control card and the results were almost identical. These kinds of cards are useful in small to medium level machines but are not feasible for large

machines and machines where there is shortage of manpower. These solutions can be used in research field and lab assignments also.

Figure 3(a) shows the beam current in the storage ring and figure 3(b) shows the lifetime of the beam stored. There are two peaks around 4.5 hours and 5.3 hours visible in figure 3(b) that shows lifetime of the beam increase suddenly. It is due to the transverse instability that blows up the beam which results in the increase in the lifetime of the beam.

REFERENCES

[1] Linux Online, "What is Linux?" web resource: http://www.linux.org
[2] "Embedded systems using linux", web resource: http://www.bamfolks.com/~randy/students/embedded/embedded_linux.html
[3] "Linux systems development on an embedded device: Tinkering with PDAs for fun and profit", web resource: http://www.techonline.com/community/ tech_topics/20704
[4] "Embedded linux opportunities: An overview", web resource: http://www-128.ibm.com/developerworks /linux/library/l-embl.html
[5] "Embedded Linux application: An overview", web source: http://www.linux-france.org/lug/ploug/doc/l-embl.pdf
[6] H. Winick, *Synchrotron Radiation Sources, A Primer*, (World Scientific Publications Co Pte Ltd, Singapore, 1995) Chapter 9, Accelerator Controls And Modeling.
[7] J.P. Scott and D.E. Eisert, *Proc. of Europhysics Conf. On Control Systems for Experimental Physics* (Villars-sur-Ollon, Switzerland, 1983), p.103
[8] http://www.arm.com/products/CPUs/index.html
[9] http://www.srsys.com/products/SIM900.htm