AN APPROACH OF RTLINUX BASED ADVANCED DATA ACQUISITION AND CONTROL SYSTEM

Manivannan.M

Department of ECE
Anna University of Technology Coimbatore, Tamilnadu, India
Email: manivgce@gmail.com

Abstract

Design of Data Acquisition and Control System (DACS) is the challenging part of any measurement and control applications. These applications widely use the single chip data acquisition method. But the single chip method has the problem of poor real time and reliability. And it doesn't able to process with embedded web documents and networked embedded devices. The main objective of this work is to overcome these drawbacks and to design a system for remote I/O data acquisition and control by Linux portable Advanced RISC Machine (ARM) processor and in-build web server application with General Packet Radio Service (GPRS) technology. This system upgrades the general single chip method. Because Linux operating system is the better option for embedded Real Time applications it doesn't have many real time constraints especially for embedded web server application. The World Wide Web is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide. This GPRS module eliminates the need of internet software suite in ARM processor and mass storage for web applications and allows the user to interface with many real time embedded applications like remote DACS and embedded processors like ARM Core.

Keywords: *Embedded ARM processor, Embedded Web Server, Linux RTOS, GPRS, Remote I/O Data Acquisition and Control System.*

I. INTRODUCTION

With the rapid development of the field of industrial process control and the fast popularization of embedded ARM processor it has been a trend that ARM processor can substitute the single chip to realize data acquisition and control. Embedded ARM system can adapt the necessary requirements of the data acquisition system, such as the function, cost, size, power consumption and so on. In this paper a new kind of embedded ARM platform has been introduced to implement efficient & high performance remote I/O data acquisition and control system (DACS) and embedded web server. This system can measure and store any kind of electrical and non-electrical signals in embedded web server. And it can able to control the devices remotely.

A. Client-Server Architecture:

A web server is a system which hosts a website and provides services for any requesting clients. The general purpose web server composes of an operating system, web pages or web applications and a huge amount of memory and sometimes a special hardware.

Fig 1. Shows the typical client-server architecture. Client accesses the industry web server through internet and LAN router. Digitally acquired data are stored in web server's data base. Whenever the client wants to access the data, it sends the request to the server; this request is taken by the router, which is connected to the internet. The web processes the request made and finally connects to the desired web server, access the requested data and sends the data to the client.

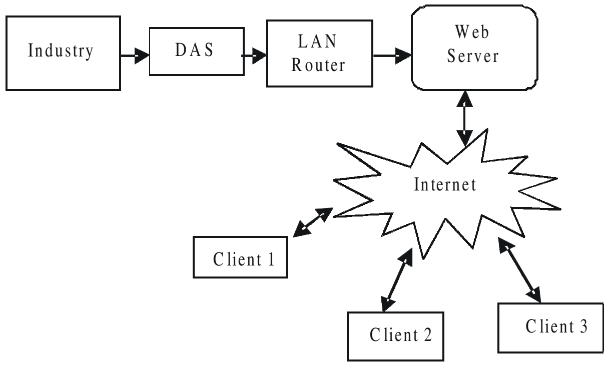


Fig 1. Client-Server architecture

B. Embedded Web Server Architecture:

General web servers, which were developed for general-purpose computers such as NT servers or UNIX workstations, typically require megabytes of memory, a fast processor, a preemptive multitasking operating system, and other resources. A web server can be embedded in a device to provide remote access to the device from a web browser. The embedded system can be utilized to serve the embedded web documents, including static and dynamic information about industry machineries/systems to web browsers. This type of web server is called an Embedded Web Server.

An embedded web server is an ARM processor that contains an internet software suite as well as application code for monitoring and controlling machines/systems. Embedded web servers are integral part of an embedded network.

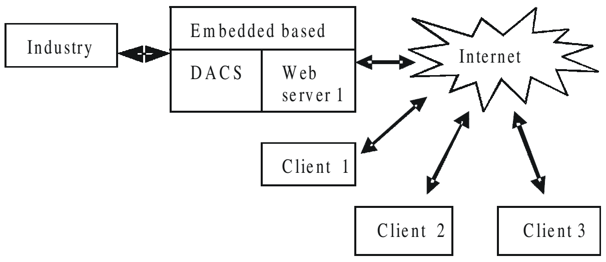


Fig 2. Embedded Web-Server architecture

Fig 2. Shows the proposed concept of DACS with embedded web server on a single chip module. This is a single hardware it contains RTOS portable ARM processor. ARM processor is the responsible part for measuring the signals and controlling the devices remotely. Measurements can be done by DACS mode and the data are shared with clients thro embedded web server by embedded web server mode. The real time operating system manages all the tasks such as measuring signals, conversion of signals, data base up-dation, sending HTML pages and connecting/communicating with new users etc.,

The RTOS manages all the required tasks in parallel and in small amounts of time. Web based management user interfaces using embedded web server have many advantages: ubiquity, user-friendly, low-development cost and high maintainability. Embedded web server has different requirements, such as low resource usage, high reliability, security, portability and controllability for which general web server technologies are unsuitable.

II. THE HARDWARE DESIGN OF THE SYSTEM

A. DACS Design

The general hardware structure of the remote I/O data acquisition and control system based on ARM processor is shown in Fig 3. The remote I/O data acquisition and control system based on embedded ARM platform has high universality, each acquisition and control device equipped with 24-way acquisition/control channels and isolated from each other. Each I/O channel can select a variety of electrical and non electrical signals like current, voltage, resistance etc., Digital acquisition are done by special ADC. The measured data are stored in external memory in which the memory is act as a data base during web server mode. The ARM processor directly supports the Ethernet service and RS485 communication. Hence the data has been stored and controlled by some other PCs or network via RS485 & Ethernet. ARM processor has internal I2C module. So it has the ability to communicate with any other peripherals.

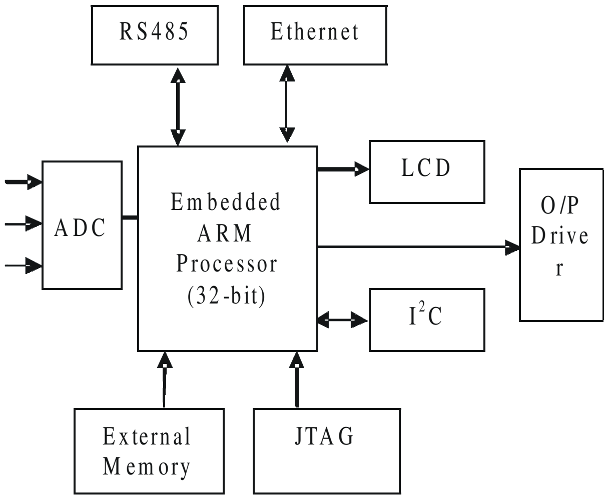


Fig 3. General Structure of the Remote I/O DACS

The I^2C is the wired communication protocol to communicate with other processor or peripherals thro wired link. This system has 128^*64 *LCD* to display the information and measured parameters which makes the debugging and modification of the parameter easy.

The Analog to digital interfacing module is independent with the embedded system, which is beneficial to the system maintenance and upgrade. As the embedded Ethernet interface makes the remote data exchange between the applications become very easy.

B. Analog to Digital Conversion

The Analog and digital conversion chip circuit is shown in Fig 4.

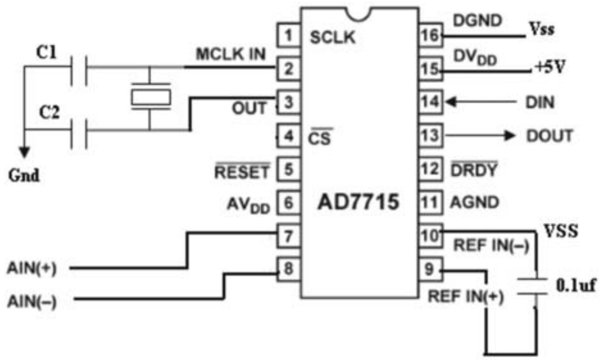


Fig 4. A/D Conversion Chip Circuit

This circuit uses the 16bit ADC chip AD7715. This is digital chip having I^2 C module internally. It has the ability to transfer the converted digital data to ARM processor. It needs only five lines, which are DOUT – Data output, DRDY – Data ready, DIN – Data Input, CS – Chip select and SCLK – system Clock. When the ADC chip has been selected by the ARM Processor then the chip start to convert the given analog signal into digital. Converted digital data will be sending out by DOUT pin of the chip. This ADC chip is driven by 2.4576MHz crystal. It contains separate Reference signals Ref+ and Ref- and separate Analog input channels AIN + and AIN – . During communication with ARM processor this ADC chip should be synchronized with the processor's clock.

C. RS485 Communication

RS-485 is a telecommunications standard for binary serial communications between devices. It is the protocol or specifications that need to be followed to allow devices that implement this standard to communicate with each other. This protocol is an updated version of the original serial protocol known as RS-232. While the original RS-232 standard allowed for the connection of two devices through a serial link, RS-485 allows for serial connections between more than 2 devices on a networked system.

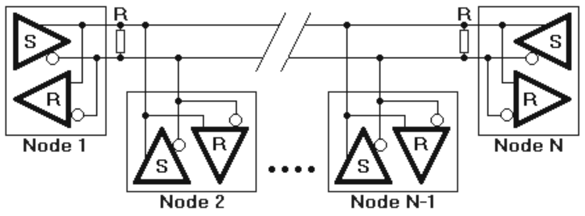


Fig. 5. Network topology of RS485

The general network topology of RS485 is shown in Figure 5. Here N nodes are connected in a multipoint RS485 network. For higher speeds and longer lines, the termination resistances are necessary on both ends of the line to eliminate reflections. Use 100 ? resistors on both ends. The RS485 network must be designed as one line with multiple drops, not as a star. RS-485 standard specifies up to 32 drivers and 32 receivers on a single (2-wire) bus. In this DACS system the RS485 communication is used to transfers the data between remote DACS to Embedded controller vice versa. New technology has since introduced "automatic" repeaters and high-impedance drivers and receivers such that the number of drivers and receivers can be extended to hundreds of nodes on a network. RS-485 drivers are now even able to withstand bus contention problems and bus fault conditions.

III. THE SOFTWARE DESIGN OF THE SYSTEM

A. RTLinux Introduction

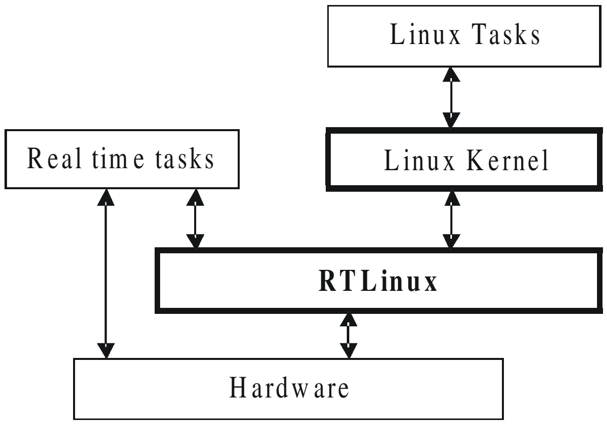


Fig. 6. RTLinux Run time Model

Unlike Linux, RTLinux provides hard real-time capability. It has a hybrid kernel architecture with a small real-time kernel coexists with the Linux kernel running as the lowest priority task. This combination allows RTLinux to provide highly optimized, time-shared services in parallel with the real-time, predictable, and low-latency execution. Besides this unique feature, RTLinux is freely available to the public. As more development tools are geared towards RTLinux, it will become a dominant player in the embedded market.

RTLinux is a typical dual-kernel, one is Linux kernel, which provides various features of general purpose OS, other one is RTLinux kernel, which support hard real time capability. Fig 6 illustrates the RTLinux architecture.

The RTLinux clock control APIs:

int init(struct rtl clock *c) ;

void uninit(struct rtl clock *c) ;

hrtime t gethrtime(struct rtl clock *c) ;

int sethrtime(struct rtl clock *c, hrtime t t) ;

int settimer(struct rtl clock *c, hrtime t interval) ;

int settimermode(struct rtl clock *c, int mode) ;

void handler(struct pt regs *r) ;

Timer manages related API:

int timer create(clockid t clock id, const struct sigevent *signal specification, timer t *timer ptr)

int timer gettime(timer t timer id, struct itimerspec *ts set)

int timer settime(timer t timer id, int flags, const struct itimerspec *new setting, struct itimerspec *old setting)

int timer delete(timer t timer id) ;

RTLinux interrupt API:

add/remove real time interrupt handlers

int rtl request irq(unsigned int irq, unsigned int (*handler)(unsigned int irq, struct pt regs *regs));

int rtl free irq(unsigned int irq);

install/remove software interrupt handlers

int rtl get soft irq(void (*handler)(int, void *, struct pt regs *), const char * devname);

void rtl free soft irq(unsigned int irq);

B. RTLinux Development

Fig 7. Shows RTLinux System Architecture Developing real-time applications in RTLinux usually require splitting the application into two parts:

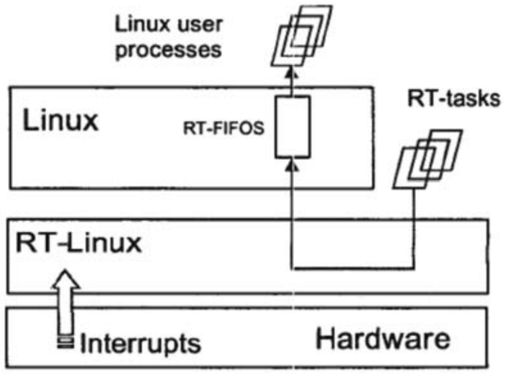


Fig. 7. RTLinux System Architecture

RT-tasks *or* RT-threads: tasks with hard real-time constraints programmed and executed at kernel level.

Linux *user* processes: tasks with soft or no real-time constraints. Unlike RT-tasks, the graphical system and some networking facilities are available for these tasks, so those parts of an application that need these resources have to be executed as Linux processes.

Both parts of an application can communicate using a special device called RTFIFOs. The execution of RT-tasks is done using the Linux facility to dynamically load new kernel modules. RT-tasks are dynamic kernel modules. Even RTLinux is a set of kernel modules (scheduler, fifos...) that need to be dynamically loaded.

C. TCP/IP in RTLinux

LWIP is an implementation of the TCP/IP stack "The focus of the LWIP stack is to reduce memory usage and code size, making LWIP suitable for use in small clients with very limited resources such as embedded systems". Improvements achieved by LWIP in terms of processing speed and memory usage have been performed by means of violating the TCP/IP layers. Most TCP/IP implementations keep a strict division between the application layer and the lower protocol layers. As the barrier between the kernel and the application processes is not a strict protection, a more relaxed scheme for communication between the application and the lower layer protocols can be performed by means of shared memory. In particular, the application layer can be made aware of the buffer handling mechanisms used by the lower layers. Therefore, the application more efficiently reuses buffers. Also, since the application process can use the same memory as the networking code the application can read and write directly to the internal buffers, thus saving the expense of performing a copy. As in many other TCP/IP implementations, the layered protocol design was used as a guide for the LWIP design and implementation. Each protocol is implemented as its own module, with a few functions acting as entry points into each protocol. Even though the protocols are implemented separately and as said before, some layer violations are made in order to improve performance both in terms of processing speed and memory usage. For example, when verifying the checksum of an incoming TCP segment and when de multiplexing a segment, the source and destination IP addresses of the segment has to be known by the TCP module. Instead of passing these addresses to TCP by the means of a function call (and performing a copy), the TCP module is aware of the structure of the IP header, and can therefore extract this information by itself (thus saving the expense of a copy), All of this is internal design stuff, but for the end user LWIP provides a tailor made API that does not require any data copying. This kind of scheme makes LWIP suitable for its use in embedded systems.

In conclusion, many benefits can be obtained using

RTL-LWIP. Besides providing IPV6 and TCP protocols, RTL-LWIP is more suitable for embedded systems, not only for its code but for its memory usage improvements. Finally, other benefit inherited from LWIP is that all improvements done in LWIP can be easily ported to RTLinux. For example: the RTL-LWIP distribution has a module which is a HTTP server for RTLinux which has been ported directly from LWIP (others, like a DHCP server, can be easily ported).

D. Porting LWIP to RTLinux

LWIP consists of several modules. Apart from the modules implementing the TCP/IP protocols, a number of support modules are implemented. The support modules consist of the operating system emulation layer (sys-arch module), the buffer and memory management subsystems, network interface functions and functions for computing the internet

Checksum. Except the sys-arch module, the rest of modules are independent of the operating system, so minimal modifications have been done. Portability of LWIP stack has been achieved by gathering together all the operating system specific function calls and data structures into the operating sys-arch module, SD when any of the rest of modules need such functions the sys-arch module is used. It is important to emphasize that since the OS architecture under the sys-arch module (RTLinux) is POSIX-compliant (or at least becoming more and more POSIX-compliant).

IV. MERIT OF THE SYSTEM

A. Existing work

The use of single chip Data acquisition system (DAS) method in Instrumentation and process control application is not only limited in processing capacity and also the problem of poor real time and reliability. General web server requires more resources and huge amount of memories. This system can only measure the remote signals and it cannot be used to control the process.

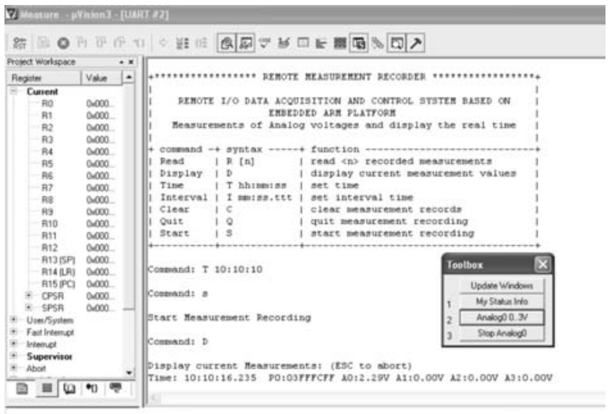
B. Proposed work

Limited processing capacity and the problem of poor real time and reliability of DAS system has been overcome by the substitution of embedded ARM processor for single chip method to realize data acquisition and control (DACS). This DACS system can able to measure the remote signals and can control the remote devices through reliable protocols and communication network. This system uses RTLinux Multi-tasking operating system to measure and control the whole process. And the embedded web server mode requires less resource usage, high reliability, security, controllability and portability.

V. CONCLUSIONS

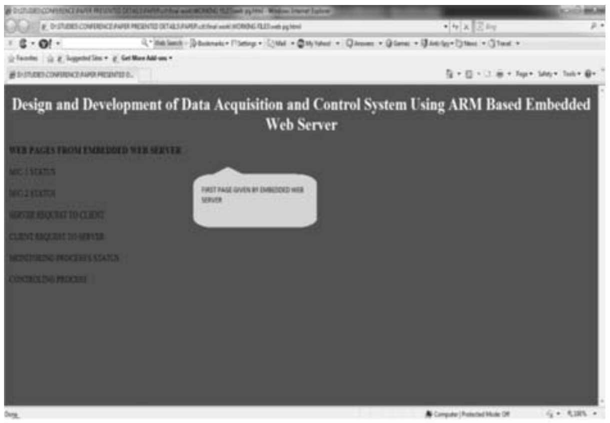
With the rapid development of the field of industrial process control and the wide range of applications of network, intelligence, digital distributed control System, it is necessary to make a higher demand of the data accuracy and reliability of the control system. This embedded ARM system can adapt to the strict requirements of the data acquisition and control system such as the function, reliability, cost, size, power consumption, and remote access and so on. This system operated by DACS mode to acquire the signals and control the devices remotely. Embedded web server mode is used to share the data with clients in online. Both modes are efficiently carried out by real time multi tasking operating system (RTLinux). This system can be widely applied to electric power, petroleum, chemical, metallurgy, steel, transportation, Electronic & Electrical industries, Automobiles and so on.

Fig 8, 9 & 10 shows the few Simulation and execution results of ARM web server based DACS system.



```
+**************** REMOTE MEASUREMENT RECORDER ****************+
|                                                              |
|    REMOTE I/O DATA ACQUISITION AND CONTROL SYSTEM BASED ON   |
|                   EMBEDDED ARM PLATFORM                      |
|   Measurements of Analog voltages and display the real time  |
|                                                              |
+ command -+ syntax -----+ function ---------------------------+
| Read     | R [n]       | read <n> recorded measurements      |
| Display  | D           | display current measurement values  |
| Time     | T hh:mm:ss  | set time                            |
| Interval | I mm:ss.ttt | set interval time                   |
| Clear    | C           | clear measurement records           |
| Quit     | Q           | quit measurement recording          |
| Start    | S           | start measurement recording         |
+----------+-------------+-------------------------------------+

Command: T 10:10:10

Command: s

Start Measurement Recording

Command: D

Display current Measurements: (ESC to abort)
Time: 10:10:16.235  PO:03FFFCFF A0:2.29V A1:0.00V A2:0.00V A3:0.00V
```

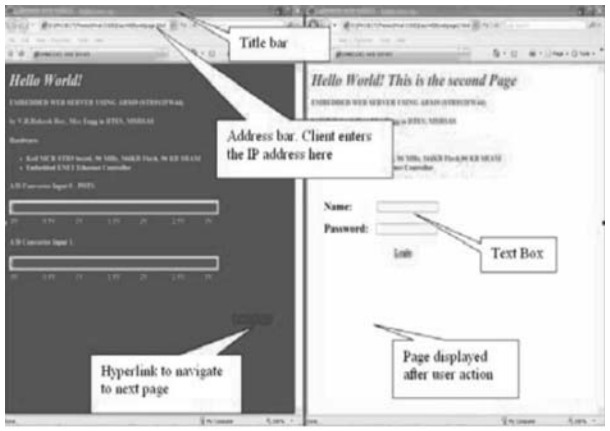
Fig. 8. Simulation result of ARM-DACS



Design and Development of Data Acquisition and Control System Using ARM Based Embedded Web Server

WEB PAGES FROM EMBEDDED WEB SERVER
FIRST PAGE GIVEN BY EMBEDDED WEB SERVER

Fig. 9. Client requested web page (Issued by ARM web server)



Title bar
Hello World!
Hello World! This is the second Page
Address bar. Client enters the IP address here
Name:
Password:
Text Box
Hyperlink to navigate to next page
Page displayed after user action

Fig. 10. On-line processing web page

REFERENCES

[1] Clyde C.W. Robson, Samuel Silverstein, and Christian Bohm" An Operation-Server Based Data Acquisition System Architecture" *IEEE TRANS ON NUCLEAR SCIENCE,* VOL. 55, NO.1, 2008

[2] PENG D.G., ZHANG H., JIANG J.N. "Design and Realization of Embedded Web Server Based on ARM and Linux". Mechatronics, 2008,14(10):37-40.

[3] Vijendra Babu D., Subramanian P. and Ravi Kanan N., "*Microblaze and Uclinux Based Data Acquisition on Spartan 3E* " Proc. Of International Conference on VLSI Design and Embedded Systems, 2008, pp1-4.

[4] Silverstein S.B., Rosenqvist J., and Bohm C., "A simple Linux-based platform for rapid prototyping of experimental control systems," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 3, pp. 927–929, Jun. 2006.

[5] Tao Lin, Hai Zhao, Jiyong Wang, Guangjie Han and Jindong Wang, "An Embedded Web Server for Equipment ", School of Information Science & Engineering, Northeastern University, Shenyang, Liaoning, China

[6] Jin M., Zhou X., and Jin L., Embedded System: components, Principles, Design and Programming, Posts & Telecom Press, China, 2007.

[7] Du Y., and Liu C., "Testing Method for Embedded Real-time System Software Control & Automation", Vol. 23, No. 4-2, 2007, pp. 86-88.

[8] Linux Kernel API, http://www.kernel.org/doc/htmldocs/kernel-api/index.html

[9] RTLinux – http://www.rtlinux.org

[10] www.fsmlabs.com

[11] www.embeddedarm.com

[12] www.opensource.org

Mr.Manivannan M is doing his P.G degree M.E (Embedded Systems Technology) in Anna University of Technology Coimbatore, Coimbatore. He has completed his B.E (ECE) degree from Government College of Engineering, Tirunelveli during 2008. In 2005 he has completed his Diploma Electrical and Electronics Engineering from K.L.N.M Polytechnic College, Madurai. He has one year Real time Industry experience in Embedded Systems. He published papers in eight National conferences and four International Conferences. His research areas include Embedded operating Systems, Kernel, RTOS, Embedded Control and Networking, and VLSI chip design.