Embedded Web Server based Interactive data acquisition and Control System using RTLinux

Mr.S.B.Kawade,
Department of E&TC, Raisoni College of Engineering, Ahmednagar (Pune University)

Prof.S.K.Waghmare
Department of E&TC, Raisin College of Engineering, Ahmednagar (Pune University)

IJERT

Abstract

Data acquisition systems (DAS) interface between the physical parameters like as temperature, pressure, flow which are analog, and the artificial world of digital computation and control. Collecting, monitoring and controlling data is a tedious and lengthy process. Although necessary, it is a task that we would rather not spend much more time on it. There are data-acquisition and control devices that will be a replace for a operator and supervisor in a various site job operation therefore it's require only single person for interact and monitoring the system. The idea behind of this work is to evaluate the Interactive data acquisition for industrial processes management systems in order to raise the industries activities in increasing energy efficiency. For this purpose are used advanced methods of data analysis and collection, monitoring and control systems. This system uses ARM9 Processor convenient with operating System is RTOS.

1. Introduction

Data acquisition and control systems are used in many different industries today in order to achieve greater productivity in our modern industrial societies. This paper approaches a new system that contains inbuilt Data Acquisition based on Web Server for Interactive Data Acquisition and Control system plays the major role in the rapid development of the Industries that presently employ such automatic systems include steel making, food processing, paper production, oil refining, chemical manufacturing, textile production, cement manufacturing, and others. It has been designed with the help of many electrical, electronic and high voltage equipments; it makes and Control system (DACS) with on-line interaction.

A similar system in provides data acquisition with no concern for remote access. In these applications, data are stored in a central server and are then served to the clients via the Internet. The client framework is in a central server and has all the applications. A person that needs to access any data must first access the server. An indirect access to the data-acquisition unit makes the system unattractive for real-time control applications, where direct interaction with the system may be required [2].

In these Papers, RTOS are designed to support multitasking. This feature is important for realtime applications .An RTOS will provide facilities to guarantee deadlines will be met alongwith it will provide scheduling algorithms in order to enable deterministic behaviour in the system. Also RTOS is valued more for predictability than throughput .During task scheduling, a context switch is sometimes described as the kernel suspending execution of one process on the CPU and resuming execution of some other process that had previously been suspended. Its advocates also claim that software context switching allows for the possibility of improving the switching code [9].Embedded computer technology, as an important part of computer field, is closely related to people's lives and has become hot in research and application area. Data acquisition, which is an important branch of computer applications, is an integrated application of technology, based on sensors, signal measurement, data processing and embedded systems. Because of a wide variety of signals of measured object e.g. electrical parameters of current, voltage, power, frequency, analog, circuit breaker status, digital signal protective actions, the amount of electrical pulses degree, non-electrical Parameters of temperature, pressure and other thermal signal, water level , flow and other hydraulic signal; pulse, ECG, speed and other signals, the usual practice is to design special data acquisition systems according to different signals, and thus, there are some limitations. Therefore, we propose to build a universal data acquisition system consisting of hardware and software platforms, based on ARM9 microprocessor core. We just need to choose different data acquisition boards according to different measured signals, and then a user-defined data acquisition system can be formed conveniently and flexibly [9].

2. Overall System Design

An Overall Interactive Data Acquisition System is comprised of two parts; Hardware (an I/O sub-system, a host computer) and the controlling software.

A) Hardware Design Section

I) IDACS Design IDACS design is the major part in hardware. ARM9 processor is a heart of this system. The general hardware structure of the IDACS is shown in Fig.1 The interactive intelligent data acquisition and control system based on embedded ARM platform for globalization, each acquisition and control device equipped with 12-way & 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, and resistance etc., Here Special ADC required for digital data acquisition. It collects the measured data and it's stored in external memory. This memory is useful for data base in Web server mode. The ARM processor handles the Ethernet service and RS485 communication. For controlling the data by some other PCs or network via RS485 & Ethernet.ARM processor has internal I2C module has supporting to the any other peripherals. Acquired data usually needs to be transferred for further processing. There are two major kinds of data transfer types: wired, wireless and software. Usually data transfer speed is not important for acquisition systems, because there is no large amount of data involved. Wire and wireless data transfer is used to transfer data from hardware sensors. Both raw and processed data can be transferred for collection. It is possible to use different types of data transfer methods in one system. Software links are used to collect data from other, already installed systems, such as SCADA, using standard data transfer protocols, such as OPC and DATA socket [5].

IJERT

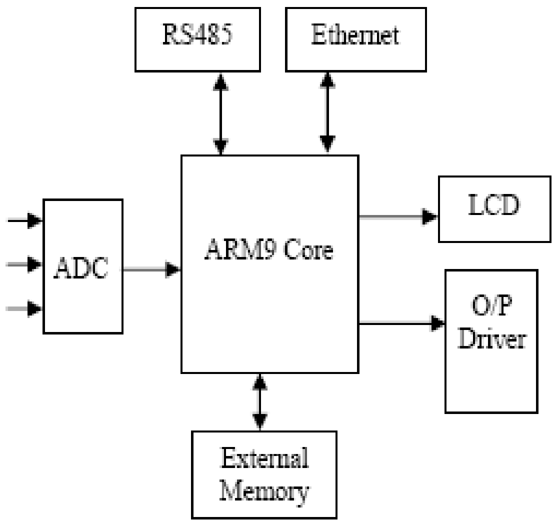

Figure. 1. General Structure of the IDACS

II) Physical I2C Bus This is just two wires, called SCL and SDA. SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. The SCL & SDA lines are connected to all devices on the I2C bus. There needs to be a third wire which is just the ground or 0 volts. There may also be a 5volt wire is power is being distributed to the devices. Both SCL and SDA lines are "open drain" drivers. What this means is that the chip can drive its output low, but it cannot drive it high. For the line to be able to go high you must provide pull-up resistors to the 5v supply. There should be a resistor from the SCL line to the 5v line and another from the SDA line to the 5v line. You only need one set of pull-up resistors for the whole I2C bus, not for each device. The value of the resistors is not critical. I have seen anything from 1k8 (1800 ohms) to 47k (47000 ohms) used. 1k8, 4k7 and 10k are common values, but anything in this range should work OK. I recommend 1k8 as this gives you the best performance. If the resistors are missing, the SCL and SDA lines will always be low - nearly 0 volts - and the I2C bus will not work [5].

B) Controlling Software of the System

I).RTLINUX Overview

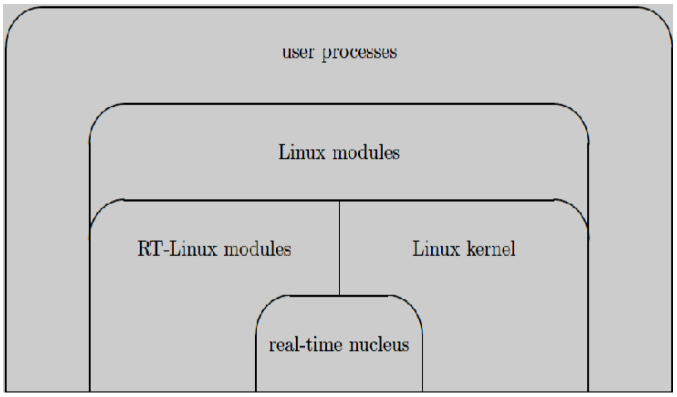

Figure. 2. RTLinux architecture

RTLinux is a hard real-time OS, as opposed to soft real-time systems or those that make best efforts toward fulfilling requests. RTLinux is a hard real-time kernel that guarantees hard real-time performance and runs Linux or BSD UNIX as an idle task when there are no real-time demands. The dual kernel approach taken by RTLinux requires a slightly different approach to real-time programming. Real-time code exists as threads managed by the real-time kernel. All user management code is run as a normal process managed by Linux, communicating through a variety of mechanisms, such as FIFOs and shared memory. This code separation abstracts real-time code into a simpler code base concentrating solely on real-time demands, simplifying development of both the real-time code and the management interfaces. Some real-time OSs approaches try to force the kernel and user space code to do well with both real-time and non-real-time scheduling constraints, often with complex and disastrous results. Instead, RTLinux takes the normal UNIX approach, where a tool is written to do one thing and do it well, rather than cramming everything into a 'one size fits all' system. This results in a simpler system encouraging simpler code, while simultaneously providing a deterministic real-time environment that is constrained only by the hardware powering it [4].

IJERT

II) Interrupt handling Interrupts usually block the highest priority tasks. It Need to minimize the unpredictability and latency. There are two types of interrupts in RTLinux: hard and soft. Soft interrupts are normal Linux kernel interrupts. Hard interrupts, on the other hand, have much lower latency.

rtl_request_irq () - Add RT Interrupt Handler

rtl_free_irq () – Remove RT Interrupt Handler

rtl_get_soft_irq () - Install Software interrupt Handler

rtl_free_soft_irq ()–Remove Software interrupt Handler

rtl_global_pend_irq () - Schedule a Linux Interrupt

THE REAL-TIME CODE:

```
#include <rtl.h>

#include <time.h>

#include <unistd.h>

#include <rtl_sched.h>

#include <rtl_fifo.h>

#include "control.h"

RTLINUX_MODULE (thread_mod);

Pthread_t tasks;

int fds;
```

A realtime application is usually composed of several ``threads" of execution. Threads are light-weight processes which share a common address space. Conceptually, Linux kernel control threads are also RTLinux threads. In RTLinux, all threads share the Linux kernel address space [4].

II) Scheduling in RTLINUX

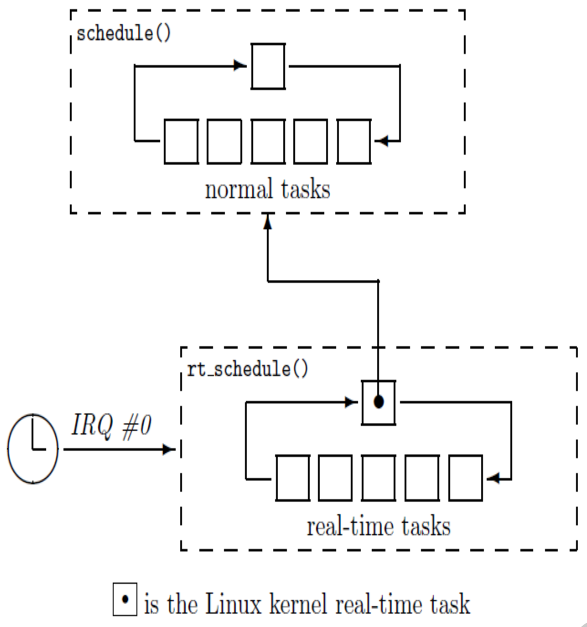

Figure. 3. RTLinux Scheduling

RTLinux can use different scheduling algorithms depending on the needs of its users. RTLinux uses a simple FIFO scheduling but can also use EDF and rate monotonic Scheduling if so desired. For compatibility reasons RTLinux makes available the POSIX function calls for scheduling [9].

3. Web Server Analysis

As an example of how to use the previously described module, a demonstration HTTP server was implemented. The module must have been powered on, properly connected to LAN and the TCP/IP settings of the local host correctly configured. Then, the embedded Web server is ready. The server provides an HTML Web page that is stored in MCU flash memory. The module waits for an incoming connection, transfers the Web page, closes the connection and waits for another client to connect. The content of this Web page is adapted dynamically with analog values. Before sending a segment of TCP data, it searches the transmit buffer for special strings. If such a string is found, it is replaced by an A/D converter value. The page has three HTML labels that display Analog-to-Digital (A/D) values such as CPU/air temperature and operating voltage and a radio button pair that toggles the main board Light Emitting Diode LED) state. One purpose of a small Web server is to make a product ease of use. This page is bidirectional in that it both displays device information data and controls the board LED on or off. The new state of the LED is sent to the Web server in a post message. The Fig.4.Shows the overall system design of Web Server [3].

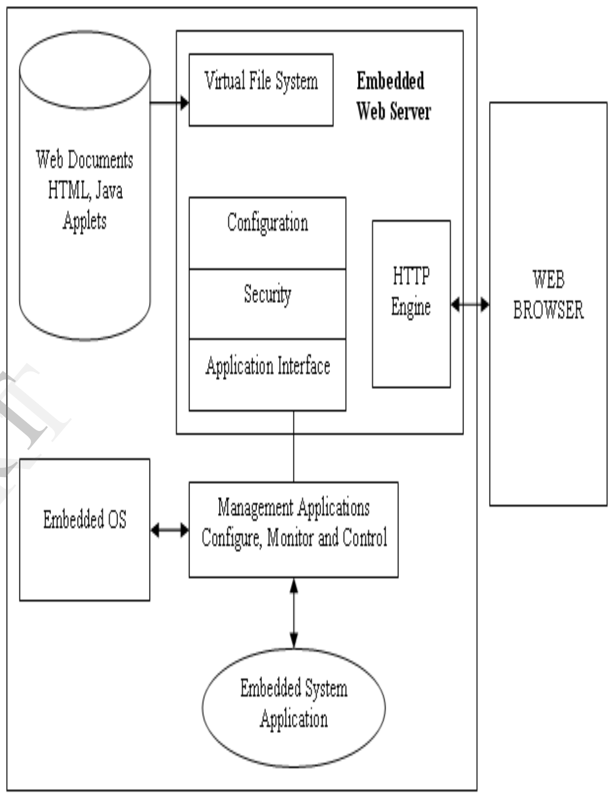
Web Documents HTML, Java Applets
Virtual File System
Embedded Web Server
Configuration
Security
Application Interface
HTTP Engine
WEB BROWSER
Embedded OS
Management Applications Configure, Monitor and Control
Embedded System Application

Figure. 4. Overall System Design of Web Server

4. Future Scope

Future research is needed to improve performance. A final consideration of this work is that there is a very rich field involving the choice of the most suitable RTOS for maybe critical or non critical embedded tasks, it is necessary to make a testing the system under an overload situation, i.e. the system is presented with more transactions than it can handle. Also testing the Real -time applications have the requirement to meet task deadlines in addition to the logical correctness of the results. This embedded ARM system can adapt to the strict requirements of the data acquisition and control system. In these System, both modes are

IJERT

efficiently carried out by real time multi tasking operating system (RTLinux).This system can be widely applied to chemical industry, steel making plant, pharmaceutical industry, sugar factory and others[7].

4. References

[1] Manivannan M, Kumaresan N "Design of On-line Interactive Data Acquisition and Control System for Embedded Real Time Applications", Measures OF ICETECT 2011.
[2] M Poongothai "Design ARM Embedded Web Server Based on DAC System", IEEE Trans, Jan. 2011.
[3] Wang Jiannong, Wang Wei "The Common Data Acquisition System Based On Arm9", IEEE Trans, March 2011
[4] "FSM Lab Mannual".
[5] Kyasa Shobha Rani, Prof. B.Bramha reddy "Design of On-line Interactive Data Acquisition and Control System for Embedded Real Time Applications",IJRCCT,ISSN 2278-5841,Vol. 1,Issue 6, Nov.2012.
[6] "A Survey of Real-time Operating Systems"
[7] Shahmil Merchant, Kalpen Dedhia "Performance Comparison of RTOS".
[8] www.fsmlabs.com.
[9] Benjamin Ip "Performance analysis of VXWorks & RTLinux" COMS W4995-2, Languages of Embedded Systems Department of Computer Science, Columbia University.

IJERT