TELKOMNIKA, Vol. 11, No. 4, April 2013, pp. 1990~1995
ISSN: 2302-4046

Research and Design of Embedded uC/OS-II Network Storage System

Xiaowei Zhang*[1], Fensu Shi[2]
[1,2]School of Computer Science and Engineering,Beifang University for Nationalities,Yinchuan Ningxia,P.R.China.
*Corresponding author, e-mail: zxw584@163.com

Abstract

This article describes a network storage system, which is developed on the embedded system platform. It uses the open-source embedded real-time operating system kernel uCOS-II. Because the kernel is far from complete, such as the lack of a file system, device management, network protocol stack, graphical user interface, but it is small and open source.We extended its functions, such as to add LwIP network protocol stack, IDE hard drive, DM9000 NIC driver, NOR FALSH drive and a file system that supports large-capacity memory and etc, making it an embedded network storage system based on uCOS-II.

Keywords*: embedded system, uC/OS-II, LwIP protocol, network storage, device driver*

1. Introduction

With the development of micro-computers and digital information, the production of all aspects of life have contact with the network getting closer and closer, increasing demands for information access and storage space, network storage products higher requirements. The development of embedded systems is now in full swing, and a hundred schools of thought contend, and embedded systems applications together with the network are endless, the Internet of Things, network storage, remote monitoring embodies all the booming future embedded systems. Although there are many aspects of the network storage products at home and abroad, but they are commercial confidentiality, is not conducive to the research study.

This system is the combination of the embedded operating systems and network protocols, coupled with a large-capacity hard disk as a storage device, a network storage server. It is to meet people quick access and mass storage of large amounts of information, embodied energy and cost savings at the same time, to facilitate people's day-to-day office and production life.

2. System Components and the Main Function

The system hardware includes: s3c2410(ARM920T) Samsung chip, DM9000 Ethernet interface chip, AM29LV160DB NOR Flash chips, HY57V561620CT memory chips, EPM7032AE and XC95144XL hard disk interface chip and a Hitachi 120GB notebook hard drive; The software includes: uC/OS-II operating system kernel the, LwIP lightweight network protocol, the hard disk drive and modify the transplanted fat file system. Hardware components diagram shown in Figure 1.

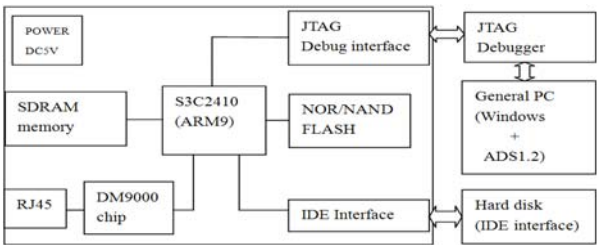

Figure 1. System hardware components

Received January 7, 2013; Revised February 19, 2013; Accepted March 2, 2013

The role of the various components of the system: the S3C2410 CPU as the system processor, NOR FLASH is used to store system code, and the hard drive is used to store files, DM9000 is used to implement the network function, DRAM is used as system memory. Function of the system diagram as shown in Figure 2.

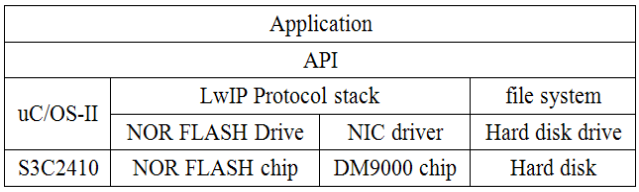

Figure 2. The system functional diagram

Main function of the system: to expand the functionality of the embedded operating system kernel uC/OS-II, let it be able to manage the files in your hard disk, transplant LwIP network protocol functions through the network upload and download hard disk file.

The key technologies of the research topic:

- uC/OS-II transplantation and expansion
- NOR/NAND Flash driven preparation
- DM9000 card chip driven preparation
- IDE interface and driven
- The modify and transplantation of the embedded file system that supports mass storage
- Lightweight TCP/IP the protocol stack LwIP of research and transplantation

3. The Design of Various Components of this System

3.1. Embedded Operating System Choice and uC/OS-II Transplanted

uC/OS-II is an open source preemptive multitasking microkernel RTOS, including an operating system basic characteristics, such as task scheduling, task communication, memory management, interrupt management, but its function is far from complete, such as the lack of a file system, device management, network protocol stack, a graphical user interface. So, in this system, we need to be transplanted and functions expansion.

3.1.1. Transplant Description

The transplantation of uC/OS-II mainly is to handle three files: OS_CPU.H, OS_CPU_C.C and OS_CPU_A.ASM.

a. OS_CPU.H mainly contains the compiler data type definitions, the definition of the stack type, as well as several macro definitions and function descriptions. Re-defined data types in order to increase the portability of the code.

b. S_CPU_C.C contains a C function with the transplant, including the realization of the stack initialization function and some hook functions.

c. S_CPU_A.ASM contains transplant-related assembly language function, open or close the interrupt, context switching, the clock interrupt service routine.

3.1.2. Porting Process

First, establish in the generic PC CodeWarrior for ADS1.2 software development platform migration projects uC/OS-II project, adding the initialization and boot code organization uC/OS-II source code, and a variety of program data. ARM executable file is generated by the project compiled, then loaded with the debugger and debug it, and finally through the JTAG interface bin file is downloaded to the onboard FLASH. uC/OS-II Kernel transplantation can successfully run offline.

3.2. The Preparation of Various Drivers

3.2.1. The NOR Flash Drive Preparation

You can use AMD chips AM29LV160DB or Intel's NOR flash chip JS28F640J3D, as the system storage chip. The chip manufacturer has its own a set of operation command set. We as long as according to the operation sequence can operate them.

Let us take AM29LV160DB as an example to illustrate chip driver write method (please refer to the Command sequence in the manual of chips Am29LV160D). If we take the word mode (To address the left one) to read and write the chip: the write operation is divided into four steps: We first wrote the data 0xaaaa to address 0x5555, then write data 0x5555 to address 0x2aaa, go down to write the data 0xa0a0 to address 0x5555, and finally the user data is written to the specified address in the chip; Read operation, as long as we read the specified address; Reset operation, we only have to write the data 0xf0f0 to address 0x0.The entire operation, we have adopted a "read / write - status query" cyclic operation. It should be noted that the chip in writing before the erase operation, because the FLASH writes only the corresponding bit from 1 to 0, and erase to block bit from 0 to 1.

3.2.2. DM9000 Card Chip Driver

The DM9000 card chip occupies s3c2410 the bank1 space, its I / O address space is 0x0a000000 + 0x300 address port: 0x0000, data port: 0x0004. According to the DM9000 data manual to write drivers for the DM9000 to achieve basic operations: ReadReg, WriteReg, IOREADw, IOWRITE, dm9000Probe, dm9000Reset, dm9000Init, Enable_dm9000_irq , DM9000DBG_IsReceivedPacket, RcvPkt, TransmitPkt. We can according to DM9000 data manual instructions to write DM9000 driver, to achieve the basic operation are: ReadReg, WriteReg, IOREADw, IOWRITE, dm9000Probe, dm9000Reset, dm9000Init, Enable_dm9000_irq, DM9000DBG_IsReceivedPacket, RcvPkt, TransmitPkt. These functions can be placed in a separate file lib_emac.c, can also be embedded directly to LwIP interface file ethernetif.c of call. It should be noted that DM9000 driver preparation should be closely linked with the uC/OS-II kernel and LwIP network protocol.

3.2.3. The Preparation of the Hard Disk Driver

Because the s3c2410 itself does not directly support the IDE interface, so we must extend the interface for it. This system uses a Xilinx company CPLD chip XC95144XL extend the IDE interface. This CPLD chip is programmed by VHDL language via the JTAG port. The voltage of the interface chip XC95144 is 3.3V, so we must do the conversion from 3.3V to 5v between the chip and the IDE interface Used 74LVC245 chip. The connection between the chip and the IDE interface circuit diagram is shown in Figure 3:

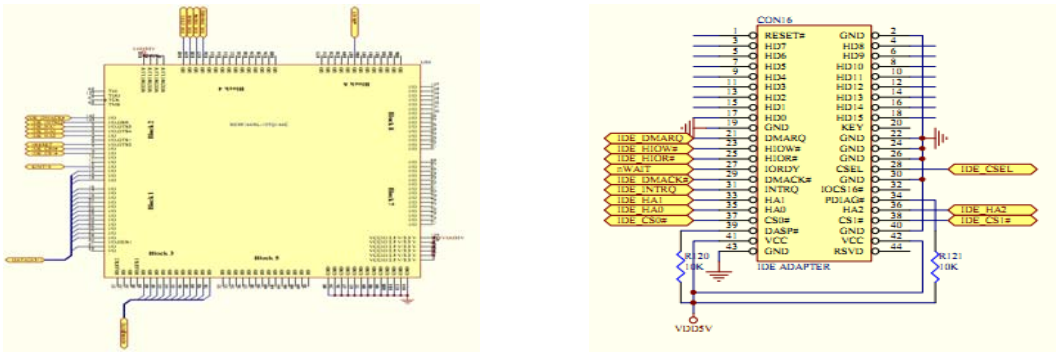

Figure 3. The XC95144XL Chip and IDE Interface Circuit

Hard disk driver: IDE interface hard disk control is achieved through two sets of registers on the hard disk controller. One group of command register groups, I/O port address is 1F0H~1F7H, whose role is to transmit commands and command parameters; another group is the control/diagnostic register, I/O port address is 3F6H~3F7H, whose role is to control the hard disk drive. The IDE interface has two data transmission modes: programmable I / O (PIO) mode

and DMA mode. In the PIO mode, CPU access to the controller through the PIO. In the DMA mode, the CPU has issued a DMA request, and complete the transfer data directly between the memory and peripherals, and then the memory access is returned to the CPU. In this system, we use the PIO mode to operate the hardware, if you want to use DMA mode, will have to use the bus decoding chip EPM7032AE for IDE_DMARQ, and IDE_DMACK the timing conversion.

3.3. Development of Large-capacity Hard Disk File System

File system is used to a certain format to store and manipulate certain data on the media. It contains two elements: a set of methods that manipulate the data and the storage structure on the media, the former refers to the file operations such as add, delete, modify, the latter refers to certain file formats stored on a storage medium. Therefore, the design of a file system should first planning good storage format on a storage medium, and then write the file CRUD operations function.

We can transplant a fat file system on the basis of study of the fat file system, you can also modify it. Fat12 file system does not support large hard drives and large files, so we refer to the fat file system to write a simple file system.

A simple file system generally requires the following elements:

1. boot sector
2. recording sector usage region
3. Record any file information, such as taking up which sectors
4. storage area of the document indexing

The file system involves manipulation functions: Hard disk initialization function, media formatting for certain file system formatting function, file creation, delete, modify and file read into memory relates to the core data structure of the process table, file descriptor table and node table.

3.4. Lightweight Embedded TCP / IP Protocol LwIP

LwIP is a small independent implementation of the TCP/IP protocol suite that has been developed by Adam Dunkels at the Computer and Networks Architectures (CNA) lab at the Swedish Institute of Computer Science (SICS). LwIP can be ported to run on the operating system, in case there is no operating system can also be run independently. The focus of the realization of LwIP is to reduce RAM usage while maintaining the main features of the TCP protocol. It generally only about tens of KB of RAM and 40KB ROM can run, which makes LwIP protocol stack suitable for use in low-end embedded systems.

The characteristics of LwIP are as follows:

(1) IP forwarding support in the case of a multi-network interface
(2) Support the ICMP protocol
(3) Experimental extended UDP (User Datagram Protocol)
(4) Including congestion control, RTT estimation and fast recovery and fast-forwarding TCP (Transmission Control Protocol)
(5) The specialized internal callback interfaces (Raw API) used to improve application performance
(6) Choice of Berkeley Interface API (multi-threaded case)

LwIP implementation of receiving a data packet and transmits a packet function framework, these two functions are low_level_input and low_level_output. users need to use the actual card driver to complete the two functions. A typical LwIP application system includes three processes: the application process, the LwIP protocol stack, the underlying hardware to process data packet receiving process. Let's analyze the interface file, take a look at how the data packet is received, as shown below:

To send packets, the upper direct call function net->linkoutput, which has been pointed to low_level_output when the network card initialization, The function low_level_output is associated with specific NIC real packet send function. Figure 5 shows the processing flow of ARP, ICMP, UDP and TCP in LwIP

LwIP contain multiple modules, in addition to the TCP / IP protocol suite (IP, ICMP, UDP, TCP) module, and also some other support modules such as: operating system emulation layer, buffer and memory management subsystem, network interface functions, and calculation of the test and the function and some other abstract API.

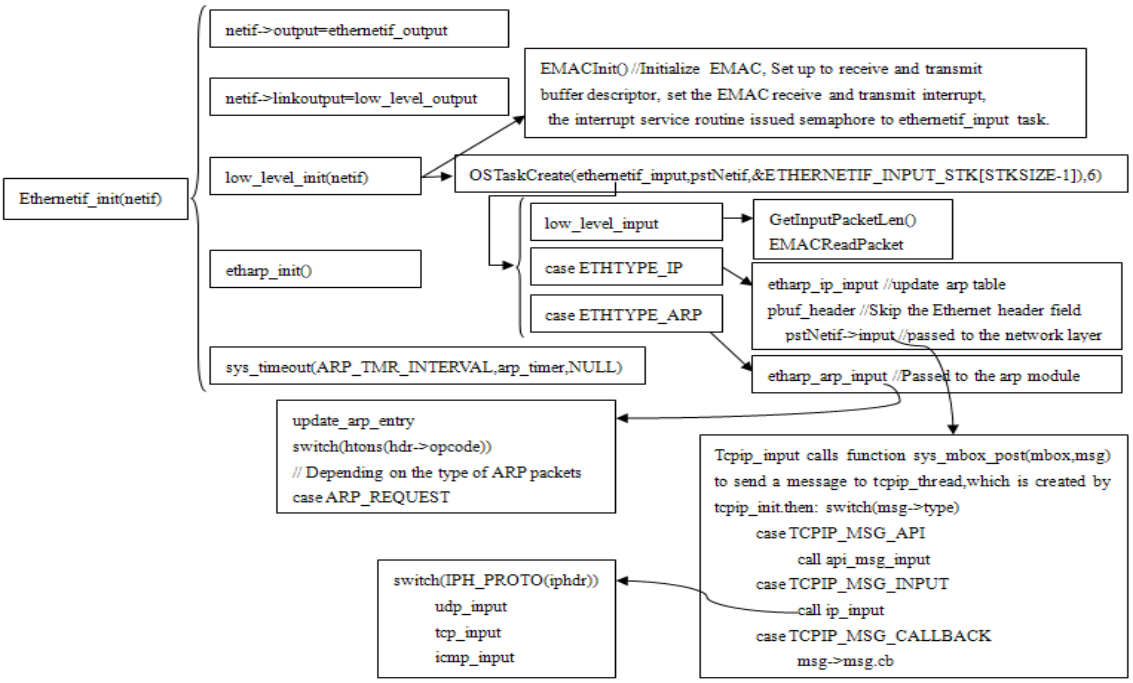

Figure 4. LwIP Network Interface Function Process

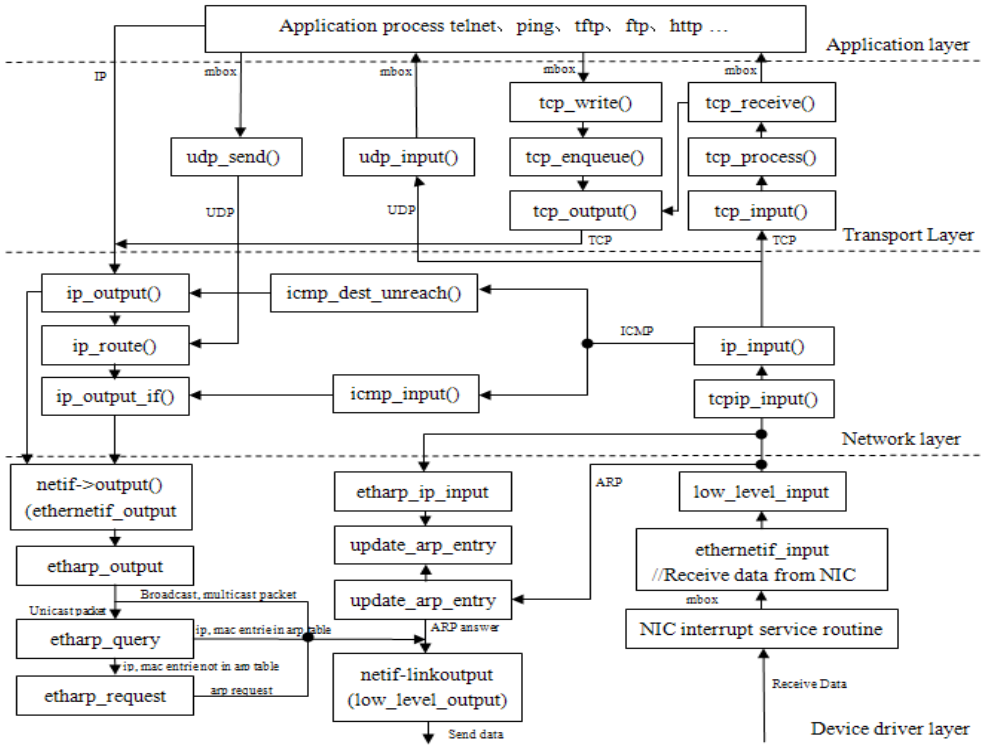

Figure 5. LwIP Upper Layer Protocol Processing Flow

4. Application Examples and Summary

Finish the work of the above, you can initialize LwIP the in uC/OS-II, and create a test mission based on TCP or UDP. Initialization of LwIP must be in the task of uC/OS-II.

```
main()
{
    OSInit();
    FS_Init();
    OSTaskCreate(LwIP_init_task, 0, &LwIP_init_stk[0][TASK_STK_SIZE-1], LwIP_init_Prio);
    OSTaskCreate(usr_task, 0, &usr_stk[1][TASK_STK_SIZE-1], usr_Prio);
    OSStart(); //start never return
    FS_Exit();
    return 0;
}
```

Hardware, kernel, file system and protocol stack initialization work should be completed in the main program. The main thread tcpip_thread and tcpecho_thread of LwIP are also created in LwIP_init_task task. Compile and run, we can get ICMP response after ping ip, can log on using telnet ip port or ftp. We can achieve a simple ftp for file upload or download on your hard disk file system, and remote management of files.

This article describes the main technical and practical experience of embedded network storage system based on uC/OS-II. The system also allows us insight into the uC/OS-II the deep operational mechanism of the operating system kernel and the function expansion method. Enhance our understanding and grasp of file systems, network protocols, drivers. It can help us to be more profound understanding of the network server, web and website, database and file servers, and embedded networking products and network storage method works. And the system has strong practical significance, individuals and companies can achieve fast access to large amounts of data through the network, it is a viable network storage solution for embedded systems.

References

[1] Haitao Zhu, Ruijun Zhu. *Design and Application of Embedded Network Storage System*. Dalian University of Technology. 2010.
[2] Chun-Lei Du. *ARM architecture and programming. First edition*. Beijing: Tsinghua University Press. 2003: 262-287.
[3] Jean J Labrosse. *MicroC/OS-II: The Real-Time Kernel*. Beibei Shao: Beijing University of Aeronautics and Astronautics Press. 2002.
[4] Adam Dunkels. LwIP source code
[5] ZhuHong Liu, Yuelong Zhao. *IDE interface driven design of embedded systems, key technology*. South China University of Technology. 2007.
[6] Cangfeng Ding, Ningjing Xue, Maolin Lu, Design and research on embedded web server. *Computer Engineering and Design*. 2009; 30(18): 4188-4191.
[7] Wilson YH Wang, Tow Chong Chong. An Ethernet based data storage protocol for home network. *IEEE Transactions on Consumer Electronics*. 2004; 50(2): 543-551.
[8] Microsoft Corporation, FAT:General Overview of On-Diskormat(V1.03). 2000.
[9] Ahmad Ashari. Distributed Monitoring and Controlling Using Microcontroller and Virtual Internet. *Telkomnika*. 2010; 8(3): 285-292
[10] Hermawan, Riyanarto Sarno. Developing Distributed System with Service Resource Oriented Architecture. *Telkomnika*. 2012; 10(2): 389-399