ISSN: 2234-991X, Vols. 6-7, pp 941-947
doi:10.4028/www.scientific.net/AEF.6-7.941

Scheme Design for Communication between Network Port and Serial Port Based on ARM

Zhen Xiaoqiong[1,a], YAO Zhendong[1,b], WANG Wenbin[2,c]

[1]Chengdu University of Information Technology, CMA Key laboratory of Atmospheric Sounding, Chengdu, Sichuan, 610225, China

[2]Operation Department Ertan Hydropower plant Panzhihua Sichuan China, Chengdu, Sichuan, 610225, China

[a]xiaoiqong.zhen@foxmail.com, [b]dspyzd@cuit.edu.cn, [c] wangwenbin@ehdc.com.cn

Keywords：ARM; network port; TCP/IP; serial port; Internet; remote control

Abstract. The data transmission scheme between a transmission control protocol/internet protocol (TCP/IP) network port and a serial port based on an ARM embedded system is introduced in the paper. The data transmission between a personal computer (PC) and remote equipment through an Internet is realized. The structure of a hardware platform is simply introduced, the cutting and transplant of operating system, the communication program design of the network port and the serial port and the cross compiling between the two ports are introduced in detail. The AT91SAM9261 chip of ATMEL Corporation is employed as the embedded board of a kernel processor. The Redhat9 Linux operating system is employed as a software development platform, an ARM-Linux operating system is employed as a program operation platform and the kernel version is Linux-2.6.2. The C program is used and an ARM-Linux-GNU tool chain is used for the cross compiling. The experimental result proves that the real-time, parallel full duplex data transmission between the network port and the serial port is realized by the scheme provided in the paper.

Introduction

With the development of an Internet technology and intelligent equipment, the application of the intelligent equipment remotely controlled through the Internet receives more and more attention from people. The remote control can be widely applied to a plurality of fields of urban public safety, industry safety production, environment monitoring, intelligent transportation, intelligent home, public health, health monitoring and the like, so people can enjoy more safety and relaxing life.[1] Meanwhile, because of the characteristics such as simplicity, stability and cheapness of a serial port, the serial port can be arranged in any intelligent equipment. Therefore, the realization of real-time data transmission between an Internet port which adopts a transmission control protocol/Internet protocol (TCP/IP) and the serial port means that the remote control on the serial equipment is realized by the Internet. The Scheme design for communication between network port and serial port based on ARM has wide application prospect.

System overview

System architecture

The system architecture of the design is as Fig. 1: the application program of the personal computer performs data change with ARM9 embedded equipment in an IP access way, the ARM9 embedded equipment performs the data change with other pieces of equipment through the serial port. The real-time and two-way data transmission between the PC and other pieces of equipment by processing data through the ARM9 embedded equipment.

ARM board hardware introduction

The AT91SAM9261 chip of ATMEL Corporation is selected as a kernel processor in the design.

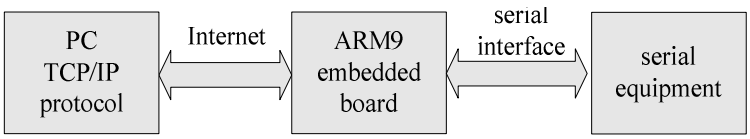

Fig. 1 System overall structure diagram

The SPX1117M3-3.3 low voltage differential regulator of Sipex Corporation is selected as a power supply part;[2] the ADM708 reset circuit of Analog Device Corporation is selected as a reset circuit part; the 28F640J3A Flash memory of the Intel Corporation and the HY57V651620BTCS SDRAM memory of the Hynix Corporation are selected as memories, and the hardware structure diagram is shown as Fig. 2.

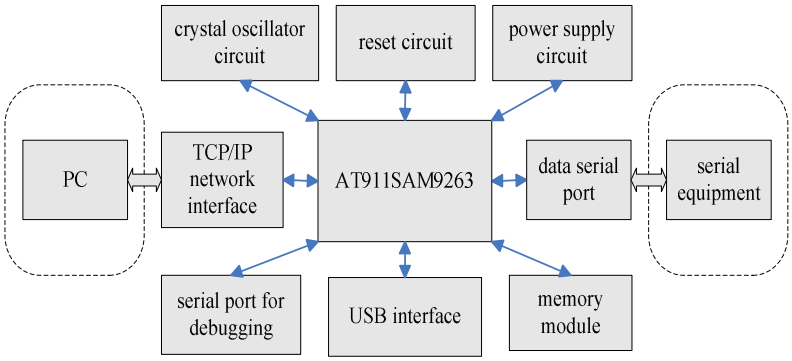

Fig. 2 Hardware structure block diagram of ARM board

Linux operating system kernel cutting and transplantation

The Linux is a mature operating system which is similar as a Unix operating system, has freedom and open source code and can be installed on various kinds of computer hardware equipment, such as a mobile phone, a tablet computer, a desk computer and a super computer. The kernel of the Linux operating system is simple, efficient and stable, can fully execute hardware functions, has more operation efficiency than other operating systems and is appropriate for applying in an embedded system.[3][4] Functions and drive which are not required in the kernel of the Linux can be cut in the application process of the embedded system so that internal memory occupied by the operating system can be reduced and the startup speed of the operating system can be accelerated.

Linux operating system kernel cutting

The kernel of the Linux operating system in the design is downloaded from an ATMEL Corporation website, the kernel is cut and compiled according to design requirement and the performance indexes of the AT91SAM9261 chip, and the specific implementation process comprises the following steps of:

(1) determining the kernel version selected in the design scheme is Linux-2.6.24 according to the requirement of system resources from network port and serial port communication program, and the download website address of the kernel is http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.tar.bz2;

(2) uncompressing downloaded linux-2.6.2 kernel: tar xvjf linux-2.6.24.tar.bz2, entering a kernel folder and waiting for the next process: cd linux-2.6.24;

(3) downloading kernel patches which are 2.6.24-at91.patch. gz and 2.6.24.at91.2-exp. Patch.gz, at website address: http://maxim.org.za/AT91RM9261/2.6/2.6.24-at91.patch.gz, uncompressing the downloaded patches to a current folder, loading the patches on the kernel: zcat linux-2.6.24-at91-exp.diff.gz | patch –p1, and the kernel copy of the Linux operating system is generated for compiling;

(4) compiling and allocating the kernel of the Linux operating system: downloading the configuration file aiming at specific chips in the design; downloading the default version of the configuration file of the chip AT91SAM9261 from the ATEML website, wherein the download

website address is http://www.at91.com/linux4sam/pub/Linux4SAM/LinuxKernel/at91sam 926yek_defconfig; entering a folder cd linux-2.6.24 where the kernel file locates; copying the configuration file to the kernel folder: cp at91sam926yek_defconfig .config; opening a configuration file oldconfig: make ARCH=arm oldconfig; performing corresponding configuration modification in the oldconfig file according to design requirements; compiling the completed configuration: make ARCH=arm menuconfig; and establishing a kernel mirror image of the Linux operating system: make ARCH= arm CROSS_COMPILE=</home/arm 2007q1 /bin / arm- none-linux-gnueabi-gcc ->. The whole process of kernel configuration of the Linux operating system is finished.

Linux operating system kernel transplantation

Kernel transplantation is realized by a bootloader program, which is the first program after the startup of the embedded system and is called Boot Loader.[5]After the startup of the system, the ROM Code with the chip completes hardware detection and resource allocation, and reads the Boot Loader to the RAM of the system, meanwhile, a system controlling right is delivered to the Boot Loader. The Boot Loader reads a kernel image from a NandFlash memory and transmits the kernel image to the RAM of the system, because the speed of the RAM memory is high and the RAM memory is more suitable for system operation, however, the RAM memory does not have a power down saving function, so the kernel of the system should be stored in the NandFlash memory before the system startup. After the system powered on, the kernel in the NandFlash memory is transmitted into the RAM memory for operating by the kernel booting program. The program skips at the entry point of the kernel for operating after the booting, and the operating system starts up.

The Boot Loader program in the paper is divided into two parts comprising an AT91BootStrap part and a U-Boot part, wherein the AT91BootStrap is started by hardware program ROM Code, then SDRAM and system clock are initialized and the U-Boot is booted to operation, finally the kernel is downloaded from the NandFlash to the SDRAM by the U-Boot, and the operating system successfully starts up. The whole flow can be shown in Fig. 3 as following.

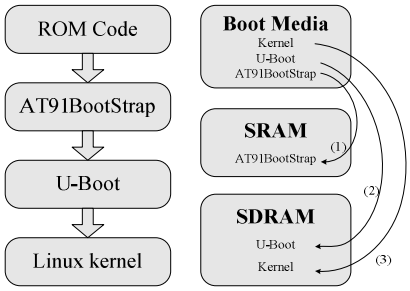

Fig. 3 System booting program flow chart

The address allocation of the Boot Loader and the system kernel are shown in Fig. 4, wherein the AT91BootStrap is a particular program aiming at AT91 series processor, and is located at an address 0x0. The U-Boot must be booted by AT91BootStrap, and is located at an address 0x20000. The addresses of the system kernel and a root file system are 0x200000 and 0x400000 respectively.

Communication application program of network port and serial port

Application program flow

The Linux operating system is a multi-process system, each process runs in an independent virtual address space, and the operating system has the characteristics such as parallelism and non-interfering.[6]

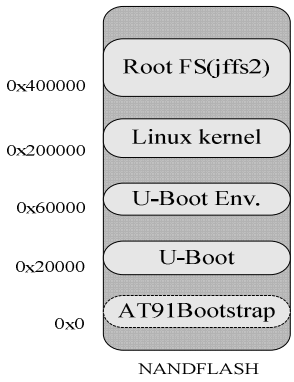

Fig. 4 Address allocation of the booting code and the kernel in NandFlash memory

The characteristics of the Linux operating system are fully utilized, the two passages such as the network port to the serial port and the serial port to the network port are realized by two processes respectively, the speed and the accuracy of data transmission are greatly improved, and the data real-time and two-way transmission is ensured. The program flow is shown as Fig. 5: after the system startup, the Linux operating system starts up automatically, and starts application programs after the corresponding hardware configuration; the application program initialize the network port and the serial port, completes the sets of Internet protocol (IP) address, a port number, a serial port baud rate and the like; the server program which is a host program of the TCP/IP is operated, server program maintains waiting and monitoring state by listen function, if the connection request from the client program, a socket connection is established, data in the socket is read and the data is stored in the buffer of the serial port, then repeat reading the data until the socket connection is over. The data transmission from the network port to the serial port is completed in this process. During the establishment of the socket connection, a host process generates another process which is a subprocess by a fork function. The generated subprocess monitors the serial port, reads serial port data circularly, and stores the data to the established socket connection until the socket connection is over. This subprocess completes the data transmission from the serial port to the network port. During the operation of the program, the host process and the subprocess run together without the influence with each other, and the real-time and parallel full-duplex transmission of the data between the network port and the serial port is realized.

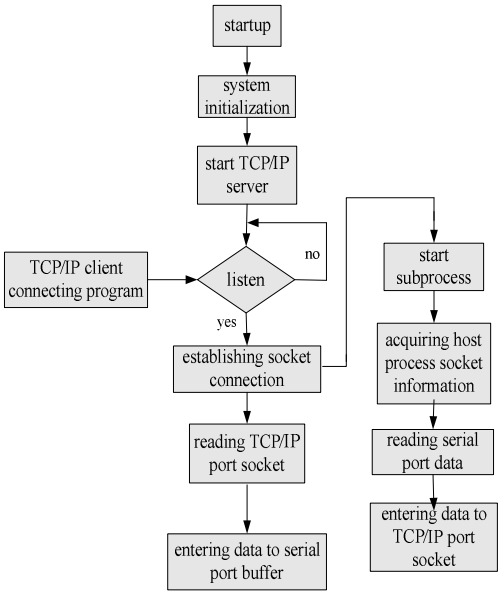

Fig. 5 Network port and serial port communication application program

Establishment between TCP/IP server and client

The establishment between the TCP/IP server and the client is realized in socket programming way. Basic functions such as socket, bind, listen, accept, send and recv are used in the socket programming in the design, wherein the socket is used for establishing the socket connection, and information such as socket type and the like can be appointed; after the establishment of the socket connection, the socketadd or sockaddr_in can be initialized for storing established socket information; the bind function is used for binding a local IP address and a port number, and other address cannot be bound; the connect function is used for establishing connection between bound client and a server; and send and recv functions are used for receiving and transmitting data after the running of connect function.

TCP protocol flow used for establishing the server and the client is shown as Fig. 6 as following.

Serial port program configuration

The AT91SAM9261 chip has two independent UART controllers,[7][8] and each controller can work in an Interrupt mode or direct memory access (DMA) mode. Meanwhile, each UART controller has 16-byte first in first out (FIFO) register, the maximum baud rate of the controllers can reach 230.4Kbps. the operation processes of the UART controller comprises the following parts such as data transmission, data receiving, interrupt generation, baud rate generation, Loopback mode, infrared mode and automatic flow control mode.

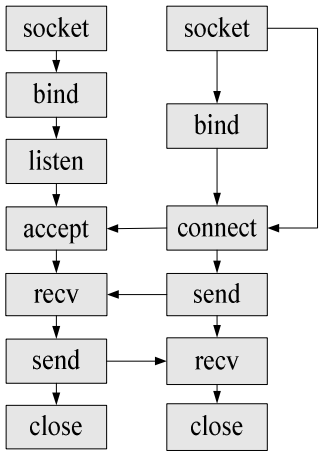

Fig. 6 TCP protocol socket programming flow chart

The serial port configuration in the design comprises the following steps[10]:

(1) activating CLOCAL and CREAD, wherein the CLOCAL and the CREAD options are used for local connection and local receiving, these two options are activated in bit mask way: Newtio.c_cflag |= CLOCAL | CREAD;

(2) setting the baud rate, which is realized by cfsetispeed and cfsetospeed functions, the baud rate in the design is 115200, and specific program is as following: cfsetispeed(&newtio, B115200); cfsetospeed(&newtio, B115200);

(3) setting character size, which is realized in bit mask way, the bit mask in the data bit is deleted at first, then the bit mask is set again, the data bit in the design is 8, and the program of this process is as following: options.c_cflag &= ~CSIZE; options.c_cflag |= ~CS8;

(4) setting parity check bit, check bit enable mark PAPRENB in the c_flag is activated, and the parity check enable in the c_iflag is activated, and the parity check is selected in the design and the corresponding program is as following: newtio.c_cflag &=~PARENB;

(5) setting stop bit, the stop bit is set by activating CSTOPB, if the stop bit is 1, the CSTOPB is eliminated, if the stop bit is 0, the CSTOPB is activated, the stop bit in the design selects 1: Newtio.c_cflag &=~CSTOPB;

(6) setting minimum character and waiting time, the receiving character and the waiting time are not particular required in the design, so the receiving character and the waiting time are set 0: Newtio.c_cc[VTIME] = 0; Newtio.c_cc[VMIN] = 0;

(7) activating configuration, after the whole serial port configuration is completed, the configuration is activated by using a tcsetattr function.

The related properties of the serial port are configured, the open, reading and writing operations can be started to the serial port, and the open, write and read functions are used.

Application program cross compiling

An executable program should be generated by compiling the application program of the network port and the serial port communication.[9] Because the program is operated on an embedded platform based on ARM, but the ARM embedded platform cannot complete the program compiling because of the storage space and the processing ability of a central processing unit (CPU). The executable program aiming to the hardware system should be compiled on other platforms, which is called the cross compiling.[10] The redhat9 version Linux operating system operated on the PC is designed as the compiling platform, the arm- 2010q1- 202- arm- none- linux- gnueabi- i686- pc- linux- gnu cross tool chain aiming at the At91sam9261 chip is adopted. The specific realization process of the cross compiling comprises the following steps of: downloading a compiling tool arm-linux-gcc-3.4.1.tar.bz2; and uncompressing the compiling tool to the folder /usr/local/arm which is realized by the instruction tar xjvf arm- linux- gcc- 3.4.1.tar.bz2. Take the compiling server program server.c as an example, the program is as following: /usr/local /arm/3.4.1/bin/arm- softfloat- linux- gnu- gcc- o server server.c, wherein the server.c is source program, and the server file is the executable file obtained after the compiling and the server can be operated on the ARM embedded board.

Executable program transplantation

The executable program is required to be transferred to the ARM9 embedded board for operating after the program compiling accomplishment. The executable program transfer method comprises the following three methods: a U flash disk direct copy, a tftp download way and setting the ARM9 board into U flash disk way. The U flash disk direct copy way is selected in the design. The specific process for transferring the server file comprises the following steps of: copying the server file to the root directory of the U flash disk; connecting the U flash disk and the USB port of the ARM9 board, and the U flash disk is automatically identified by the Linux operating system; loading the U flash disk to mnt file in the system by a mount instruction, which is completed by the program: mount –t vfat /dev/sda/ /mnt; opening the mnt folder in the Linux operating system; and reading and executing the server file.

Conclusion

Data transmission scheme between the network port and the serial port provided in the paper has been actually tested on an example template. The test proofs that the hardware system, the Linux operating system and the related application program are operated normally. The operating system core solidified in the flash can be automatically operated after the system powered on, and the application program is automatically loaded. The socket client can smoothly connect with the socket server by a preset IP address. The real time, accuracy and two-way data transmission can be realized by regulating the size of data buffer. If the serial port receiving and transmitting end in the design is replaced to the intelligent equipment having the serial port, the intelligent equipment can be remotely controlled by Internet, therefore the design scheme has wide application prospect.

Acknowledgement

The paper is supported by the two projects: public welfare industry particular project (meteorology) (GYHY200906032) and Chengdu University of Information Technology supporting project: laser intensity signal measurement method research based on CCD ((CRF201101)).

Reference

[1] Sun Qiong, Embedded Linux application program development explanation [M]. Beijing: Post and Telecom Press, 2006.

[2] Yang Guanghua, Embedded system hardware platform research based on ARM [D]. Nankai University Press, 2002.

[3] Sun Tianze, Yuan Wenju, Zhang Haifeng. Embedded design and Linux drive development guide based on ARM9 processor [M]. Beijing: Electronic Industry Press, 2005.

[4] Wayne Wolf. Components principles of embedded computing system design[M]. Beijing: Publishing House of Machinery Industry. 2002.

[5] Sun Jikun, Zhang Xiaoquan. Embedded Linux system development technology explanation based on ARM [M]. Beijing: Post and Telecom Press, 2003.

[6] Charles Crowley. Operating system oriented approach [M]. McGraw-Hill Book Co, 1997.

[7] Kurt Wall. GNU/Linux programming guide [M]. Beijing: Tsinghua University Press, 2002.

[8] Rao J.S, Zubair M, Rao C. Condition monitoring of power plants through the Internet [M]. Integrated Manufacturing Systems, 2003.

[9] Jeremy Bendlam, Chen Xiangqun (translation). Emedded system web server based on TCP/IP [M]. Beijing: Mechanical Industry Press, 2003.

[10] Hua Qing vision embedded training center. Embedded Linux C language application program design [M]. Beijing: Post and Telecom Press, 2007.