Design and development of WLAN access point based on Bluetooth and uClinux

Zhang Lei[1,2] Shen Lianfeng[1]

([1]National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China)
([2]Institute of Communications Engineering, PLA University of Science and Technology, Nanjing 210007, China)

Abstract: This paper describes the design and development of a wireless LAN (WLAN) access point based on Bluetooth and uClinux. To make the best use of high-speed serial communication capability, several methods such as modifying baud-rate of serial port driver, utilizing buffer area and adding flow-control were adopted. After analysis of scheduling and interruption handling, modifying the timer's parameters was put forward as a method to control the timer interrupt. In this way, data throughput and system stability were obviously enhanced. Meanwhile, migration of the Bluetooth protocol stack was detailed and some successful applications of our LAN access point were presented.

Key words: device driver; serial port; scheduling; interrupt handling

Bluetooth wireless technology is one of the most important new technologies in wireless LAN (WLAN) area for embedded systems development. Thousands of products are likely to incorporate Bluetooth wireless technology for short-range voice and data communications.

LAN access[1] is a traditional Bluetooth application profile among many usage models. Firstly, this profile defines how Bluetooth-enabled devices can access the services of a LAN using PPP. Secondly, this profile shows how the same PPP mechanisms are used to form a network consisting of two Bluetooth-enabled devices.

The following roles are defined for this profile.

- **LAN access point (LAP)** This is the Bluetooth device that provides access to a LAN. The LAP provides the services of a PPP server. The PPP connection is carried over RFCOMM. RFCOMM is used to transport the PPP packets and it can also be used for flow control of the PPP data stream.
- **Data terminal (DT)** This is the device that uses the services of the LAP. Typical devices acting as data terminals are laptops, notebooks, desktop PCs and PDAs. The DT is a PPP client. It forms a PPP connection with a LAP in order to gain access to a LAN. A DT uses a LAP as a wireless means to connect to a LAN. Once connected, the DT will operate as if it was connected to the LAN via dial-up networking. The DT can access all the services provided by the LAN.

Received 2003-10-24.
Foundation item: National Key Technologies R&D Program (No. 2001BA102C).
Biographies: Zhang Lei (1972—), male, doctor, lecturer; Shen Lianfeng (corresponding author), male, professor, lfshen@seu.edu.cn.

Thus, the LAP will be the key element of our project. The performance of LAP will have great impact on the whole application. So the design of our LAP is detailed in this paper.

1 Hardware Design Based on 32-bit MCU and Bluetooth Chip Set

From the point of view of hardware construction, LAP is a typical embedded system. It must embrace basic system elements such as microcontroller (MCU), RAM, and Flash. Ethernet interface and high-speed hardware channel to the Bluetooth baseband module are also required since there are protocol stacks to be executed. The hardware design is depicted in Fig.1.

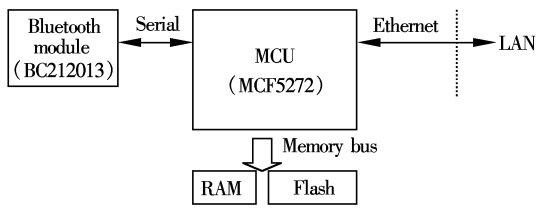

Fig.1 System hardware

The MCF5272[2] microprocessor from Motorola has been chosen as the microcontroller. It is a highly integrated ColdFire microprocessor offering a new set of communication peripherals, such as a 10/100 Ethernet controller and a USB module, but it supports popular general-purpose peripherals included on previous ColdFire standard products such as serial ports, timer and etc. Its integrated Ethernet controller meets the interface requirements of LAN while its high-speed serial port meets the access requirements of Bluetooth baseband module. Based on a Version 2 (V2) ColdFire core, it also achieves the highest V2

performance yet, with 63 Dhrystone 2.1 MIPS at 66 MHz. Thus, it is competent for both Bluetooth and IP protocol processing.

The BlueCore2-External (BC212013)[3] from CSR has been chosen as the Bluetooth baseband module. It is a single chip radio and baseband IC for Bluetooth 2.4 GHz systems. It is implemented in 0.18 μm CMOS technology with full industrial temperature operation. Its high-speed serial interface working with its counterpart in MCF5272 offers the ability to reach the maximum rate[4] (723 kbit/s) as defined in Bluetooth specification.

2 Bluetooth Protocol Stack and Software Design Based on uClinux

From the aspect of software, Bluetooth protocol stack is ready for porting while the whole system is lacking of operating system (OS) support. After evaluation on different options, a popular open-source OS — uClinux[5] has been chosen. It is used in numerous electronic devices.

uClinux2.0.38 is an open source project funded and maintained by Lineo. It was made popular by its ability to run on processors without memory management hardware and with a typical kernel footprint of only 512 Kbytes or less.

- It inherits stability and robustness from Linux;
- It supports a large number of devices, file systems, and networking protocols (especially TCP/IP);
- No license fee leads to very low cost;
- It gives developers complete visibility of the source code based on GPL license;
- Kernel can be customized on demand.

After adoption of uClinux, additional work is required to optimize the operating system. Improvement on uClinux for MCF5272 can be divided into two parts: one is rewriting serial drivers, the other is modifying parameters of the timer.

2.1 Serial driver

Since the serial port is the hardware channel between MCF5272 and BlueCore2, the performance of this driver has a great effect on the whole system. After the installation of the uClinux on our testbed, the serial driver worked normally. But it has limitations on speed and interruption of service since it was originally designed only for common use.

Firstly, minor errors in the calculation of baud-rate have been corrected according to definitions of the fractional divider from MCF5272 user manual[2]. In this way, further revision on the baud-rate table has been made so that the new serial driver not only supports common rate 115.2 kbit/s but also ultra high rates such as 921.6 kbit/s. It is this ultra high rate that can meet the peak throughput.

Secondly, the receive parts of the driver have been rewritten. With a view to compatibility, the original design issues receive interrupt to CPU after receiving one byte. But this solution results in great performance degradation in circumstance of high-speed communications for its high interruption frequency and excessive overhead in context switching. Luckily, there is a 24-byte receive FIFO in serial ports of MCF5272. If the designated fullness level of receive FIFO has been reached, serial ports can issue an interrupt signal. If the receiver FIFO has timed out at defined interval with unread data below the FIFO fullness level, serial ports will raise an interrupt signal to solve the problem of trailing data.

Thirdly, for the sake of data integrity and system stability, the mechanism of flow control to the serial driver has been added. It is much like the interrupt mechanism detailed above. If the designated fullness level of receive FIFO has been reached, serial ports can raise flow control signal to Bluetooth modules through the RTS line. In this way, it inhibits the sender from sending excessive data, which may induce overflow in hardware FIFO.

Other minor corrections involve modifying some code sequence and stripping down codes unrelated to serial ports of MCF5272.

After this enhanced driver is put into use, it has been debugged in the receive buffer among kernel space. The results show that data can be delivered from serial ports to low level drivers under an ultra high-speed condition (921.6 kbit/s). But data cannot be sent to Bluetooth protocol stack in user space normally. Therefore, the performance problem must be considered at system level.

2.2 Timer

Fig.2 shows that the data stream of serial port flows through the whole embedded system in three steps[6]. Firstly, receiving serial data byte by byte; once the defined threshold of FIFO fullness has been reached, the serial port raises the interrupt signal to CPU. Secondly, the CPU executes interrupt service routine to move data from FIFO to receiver buffer in system memory as quickly as possible. Then the CPU tags a task flag in the system for further processing before it exits the interrupt circumstance. Thirdly, when the timer interrupt has been serviced periodically, the CPU checks this task flag. If the flag has been

marked, the CPU will move data from the receive buffer to process buffer. Specific codes would be executed to do further processing on the data before the Bluetooth stack in user space obtains these data via system call.

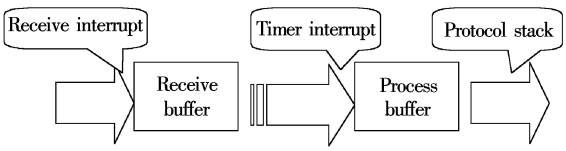

Fig.2 Data stream from UART

From the description above, analysis must be focused on the data flow between the receive buffer and process buffer since debugging shows that there is no problem in the serial driver. In view of the system scheduler[6,7], the timer interrupt becomes the impetus of further data movement. If the frequency of timer interrupt is too low, the receive buffer cannot be emptied in time. This will result in the overflow of the receive buffer and a system crash. On the other hand, if the frequency of timer interrupt is too high, the system will lack the time to deal with tasks in the process buffer. This will result in overflow of process buffer and also lead to a system crash.

After the kernel codes have been read up and the actual data flow has been traced, the overflow will appear in the process buffer. The solution of this problem is simpler than its analysis. Lowering the frequency of the timer interrupt will give enough time for the system to deal with additional tasks in the process buffer. This frequency value becomes vital to the throughput and stability because it has great effects on different parts of the whole system. After tradeoff among different tasks, actual throughput test and stability test, the frequency has been changed from 10 Hz to 5 Hz. Thus, the LAP system can work regularly under such high-speed (921.6 kbit/s).

2.3 Bluetooth protocol stack

Along with the development of the LAP, existing open-source Bluetooth protocol stacks such as Bluez, Openbt and Affix were also considered for adoption. The reasons we use our own codes instead of open-source protocol stacks are detailed below:

- Our own protocol stack has been verified. Mature products built upon this protocol stack on the PC platform have already been widely used. The performance and stability of our codes have been tested and verified. Rich control and debugging functions have been developed through our own effort. Openbt does not provide an interface to L2CAP layer. It has not been upgraded since 2001.
- Our protocol stack has been written in ANSI C. It can be easily migrated to other platforms with different OS or hardware. BlueZ stack has been contained in Linux 2.4.x and it is difficult to do reverse porting to Linux 2.0.x. Its complex networking function is not applicable to LAP.
- With respect to the intellectual property of our products, it is better to use our own codes. If these open-sourced protocol stacks were adopted, modification and enhancement of them would have to be released as a GPL license. In our realization, setting up our stack in user-space can protect the intellectual property since there is no module support in uClinux. Another benefit is that they are easily debugged.

Based on the definition of the LAP profile and discussion above, Fig. 3 depicts our solution. PPP daemon is a standard Linux program to control PPP links in user space.

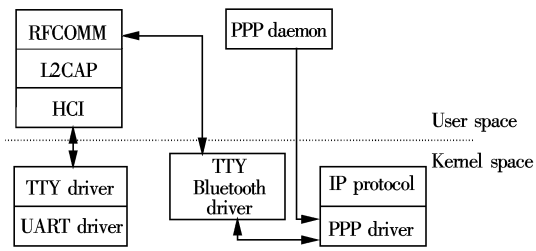

Fig.3 Realization of protocol stack UART

3 Application Examples

The standard application scenario (see Fig.4) is that multiple DTs use our LAP as a wireless means for connecting to a LAN. Once connected, the DTs will operate as if they were connected to the LAN via dial-up networking. The DTs can access all the services provided by the LAN. The DTs can also communicate with each other via the LAP.

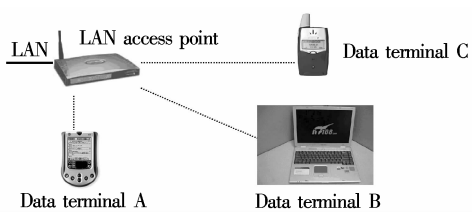

Fig.4 LAN access by multiple DTs

Furthermore, application scenarios have been extended in different fields. It has been delivered to customers in colleges as a teaching experiment platform of new communications technology. In Hong Kong, it has been used as a mature application development platform. A revised version of LAP has exhibited its high-speed capability in the project of transferring audio data point-to-point. The most successful application is that our LAPs were applied to

wireless medical treatment communication sub-system during the SARS period.

The wireless medical treatment communication sub-system is composed of Bluetooth data terminal and LAP. The structure of this sub-system is depicted in Fig.5. The practicalities of LAP and DT are depicted in Fig.6.

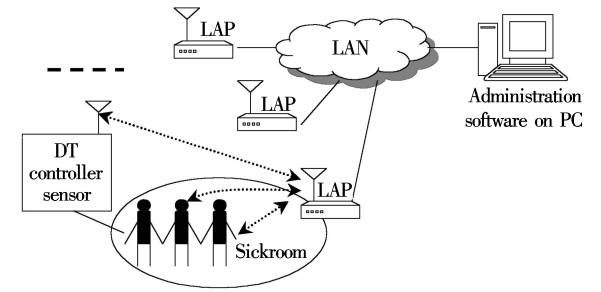

Fig.5 The structure of communication sub-system

Fig.6 Practicalities of LAP and DT

The DT, controller and sensor can be integrated in a printed circuit board, which is the same size as that of a wristwatch. Thus, patients can easily wear the "Bluetooth watch" (Fig.6). A LAP may be put near every sickroom or ward. All the LAPs are connected to the LAN via standard cable.

When the LAP is power-on, it automatically communicates with the administration PC through its LAN port. Then, it is able to initialize the Bluetooth protocol stack as quickly as possible. After initialization, the LAP enters into the state of scan in which it is ready to accept the connect request from DTs. When patients with our "Bluetooth watch" come into the area under the control of this LAP, links will be put up between LAP and DT in the "watch". After regular interaction on protocol, DT can send data such as temperature and pulse gathered from sensor upstream to administration software on the PC via LAP. The PC can also request these data downstream and gather the additional information on the location of patients from the location of the LAP.

The performance pertaining to throughput and stability is also verified in this case. This sub-system is especially useful under the circumstances of treatment with large-scale severe infectious diseases such as SARS.

4 Conclusion

The design in this paper deals with the problem of throughput and stability. After researching on drivers and system mechanism, we put forward satisfactory solutions. Firstly, to make the best use of the high-speed serial communication capability of the hardware, several methods, such as modifying interface speed of serial port driver; improving the receiving flow; utilizing the mechanism of buffer and adding the mechanism of flow control, have been adopted. Secondly, analysis has been carried out on scheduling and interrupt handling of the operating system. Modifying specific parameters solves the problem pertaining to timer interrupt that has great effect on throughput and stability.

The LAP built up uClinux has walked out of the lab and become a real product accepted by the market. The typical application cases are illustrated above. So it is proved, on the whole, that our choice of the platforms is right and the design is successful. A new design has been started along with the advance of the Bluetooth specifications and market requirements. A USB host controller can be added into the system to enrich the type of interface and improve the throughput. An enhanced version ColdFire processor MCF5282 can be selected as the microcontroller for its DMA method in serial data movement compared with the inefficient I/O method of MCF5272.

References

[1] Bluetooth Special Interest Group. LAN access profile [EB/OL]. https://www.bluetooth.org/foundry/adopters/document/9_Lanaccess/en/1/9_Lanaccess.zip. 2001-02-10/2002-04-09.

[2] Motorola, Inc. MCF5272 ColdFire ® integrated microprocessor user's manual (REV 2) [EB/OL]. http://e-www.motorola.com/files/dsp/doc/ref_manual/MCF5272UM.pdf. 2002-03-12/2003-07-08.

[3] CSR, Inc. Bluecore2-external data sheet [EB/OL]. http://www.csrsupport.com/public/83_BlueCore2-External Data Sheet (BC212015-ds-001g).pdf. 2003-03-18/2003-04-21.

[4] Bluetooth Special Interest Group. Bluetooth_core_specification_v1.2 [EB/OL]. https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2/en/1/Bluetooth_Core_Specification_v1.2.zip. 2003-06-10/2003-07-18.

[5] Yaghmour Karim. *Building embedded Linux systems* [M]. USA: O'Reilly, 2003. 30－31.

[6] Rubini Alessandro, Corbet Jonathan. *Linux device drivers*. 2nd ed. [M]. USA: O'Reilly, 2001. 269－274.

[7] Bovet D P, Cesati Marco. *Understanding the Linux kernel*. 2nd ed. [M]. USA: O'Reilly, 2002. 164－181.

基于 uClinux 和 Bluetooth 技术的 WLAN 接入设备的设计和开发

张 磊[1,2] 沈连丰[1]

([1] 东南大学移动通信国家重点实验室, 南京 210096)
([2] 解放军理工大学通信工程学院, 南京 210007)

摘要: 给出了基于 uClinux 和蓝牙技术的无线局域网接入设备的设计和研制要点.为了充分发挥硬件的高速异步串行通信能力,采用了修改串行口驱动程序中的接口波特率、利用缓冲机制并添加流量控制等方法并收到良好效果; 从分析操作系统的任务调度和中断机制入手,提出了用修改特定定时器参数的方法来控制时钟中断,该方法显著地提高了数据吞吐率,增强了系统稳定性; 同时,论述了基于 uClinux 的蓝牙协议栈的研制情况及几个关键问题,并给出了若干基于蓝牙局域网访问点的成功应用方案.

关键词: 设备驱动; 串行口; 调度; 中断处理

中图分类号: TP393.3