Design and Implementation of Multi-Serial Ports Expansion Based on ARM Embedded Linux

Yunmi Fu 1, Yiqin Lu 1, Yanhui Zeng 1, and Bin Liu 2

1School of Electronic and Information Engineering, South China University of Technology, GuangZhou, China
Email: {fuyunmi@gmail.com, eeyqlu@scut.edu.cn, yhzeng@scut.edu.cn}
2South China Household Electric Appliances Research Center, Shun De, China
Email: liub@hnjdy.com

Abstract—With the widely use of communication and intelligent devices, more and more extension modules are attached to an arm embedded system, most of them are through serial port. Thus to extend the serial ports of an arm system is necessary. This paper proposes a method of serial port expansion based on SC16C554. With SC16C554, system bus is used to extend four serial ports, which have the standard modem interface and work independently. Details of working principle of hardware and design method of device drivers of serial port expansion are presented.

Index Terms—SC16C554, S3C2410A, Serial Port Expansion, Device Driver

I. INTRODUCTION

With the increasing requirement of digital home and intelligent industrial control, embedded systems need to increase the various peripheral modules, such as the GPS module, GPRS/GSM module, ZIGBEE module, X10 module, etc [7]. These modules usually communicate with the CPU by serial ports. Because the serial interface devices have the advantages: flexible control, simple interface, occupying less resource. So they are widely used in Industrial Control, Smart Home and Prevention Technology areas [12]. But the ARM microprocessors usually provide limited serial ports, 2 or 3. One serial port is usually used for PC control, and there are only one or two serial ports to use to communicate with the microprocessor for slave devices. If a home gateway system needs to use some serial devices at the same time, such as GPS module, GPRS/GSM module, ZIGBEE module, X10 module, this home gateway system will need more serial ports resource. Therefore, the embedded system with multi-serial port expansion can be an effective solution to this problem.

II. SERIAL PORT EXTENSION METHODS

For the serial port of embedded system in the problem of insufficient, here are several common methods for serial port expansion: Software Simulation Method (SSM), Serial Port Extend Serial Port Method (SPESPM), USB Port Extend Serial Port Method (UPESM), Ethernet Port Extend Serial Port Method (EPESM) and Parallel Port Extend Serial Port Method (PPESM).

A. Software Simulation Method

SSM is based on transmission formats of serial port communication, using the timers and I/O ports of the host to simulate the serial port communication timing, in order to achieve the purpose of extending serial port. Advantages are low cost, but the reliability is poor and development of software is very difficult.

B. Serial Port Extend Serial Port Method

SPESPM usually uses the extension chip with a choice of address to extend the serial port or uses the software control timing to extend serial ports. There are two main chips of GM8123/25 and SP2338 series to use to extend the serial port base on serial to serial [5]. Advantages are simple control, occupying less resources, versatility and good stability. But the communication parameters can be less editorial, and the expanding serial ports can not work independently.

C. USB Port Extend Serial Port Method

UPESM is using a dedicated chip convert USB port to serial port. Advantages are plug, easy to expand and not needing additionally power supply, but the cost is very high, and the expansion of multi-serial port is also more complicated [5].

D. Ethernet Port Extend Serial Port Method

EPESM is using Ethernet interface to change to the serial port. The disadvantage of this method is high cost, and the design is more complex [5].

E. Parallel Port Extend Serial Port Method

PPESM is used SC16C554 to extend four serial ports which can work independently at the same time. The SC16C554 is a 4-channel Universal Asynchronous Receiver and Transmitter used for serial data communications. Through writing the control register LCR, IER, DLL, DLM, MCR, and FCR, it can achieve SC16C554 serial channels communication [1]. Baud rate generator (BRG) of serial channel allows the clock to divide any number between 1 and 65535. According to their different frequencies in one of three kinds of common standards, BRG determine the baud rate. According to regulate an external crystal, SC16C554 can

This work was supported by Guangdong Foundation of Science and Technology Project (2006A10101003, 2006A1020300, 2008B090500073), Guangzhou Foundation of Science and Technology Project (2003B11609), Project of Technology Breakthrough in Key Fields of Guangdong and Hong Kong (2006Z1)
get a very quasi-baud rate and produce many different types of baud rate from 110bps to 460800bps. Although PPESM is a little complicated to control, taking up more resources of MCU ports, such as the I/O ports, interrupt resources, but PPESM can provide MODEM control signals and expand out four serial ports which can work independently at the same time and control flexibly, communicate by high-speed and meet general serial port settings. Since we use a powerful, resource-rich S3C2410A which has 117 general-purpose I/O ports and 24-channel external interrupt source as the controlling chip, so the problem of needing many resources can be solved [2]. This paper uses this method to extend the serial ports.

III. DESIGN OF MULTI-SERIAL PORT EXPANSION INTERFACE

A. SC16C554 internal structure and working principle

The SC16C554/554D is a 4-channel Universal Asynchronous Receiver and Transmitter used for serial data communications [1]. Each channel can receive serial data from peripheral and convert them to parallel data for CPU, also can convert the parallel data from CPU to serial data and sent to the peripheral. As the interface between the CPU and the SC16C554 is based on parallel mode transmission, the interface between the SC16C554 and the peripheral is based on serial mode transmission, therefore, there must be Receive Shift Register (RSR) and Transmit Shift Register (TSR) in the block diagram of SC16C554 [1]. The SC16C554 block diagram was shown in Figure 1.

When the data from peripheral were sent to SC16C554, the data were sent to the RSR one bit by one bit. Once the RSR has received one byte data, the data from RSR were sent to the Receive FIFO Register (RFR). And then the CPU receives the data received by RFR. In the data output process, CPU sent the parallel data to Transmit FIFO Register (TFR), and TFR sent the received data to the TSR, then TSR convert the parallel data to serial data and sent to the peripheral one bit by one bit [12].

B. SC16C554 internal registers

Different combinations of address lines A0, A1, A2 of SC16C554 internal registers represent different registers. Table I details the assigned bit functions for the SC16C554 internal registers.

C. Schematic description

The circuit of Serial Port Expansion part was shown in Figure 2 and 3. SC16C554 data lines D0-D7 were connected with the CPU’s bus DATA0-7. The nIOR, nIOW lines were respectively connected with the read signal nOE line and write signal new line. The High-level RESET signal of SC16C554 was connected with the

![Figure 1. SC16C554/554D block diagram [1]](image1)

![Figure 2. The circuit diagram of SC16C554](image2)
RESET signal of CPU's hardware system. Interrupt lines INTA-C of 4 serial ports were connected with the EINT0-3 of CPU respectively, which used for sending the interrupt signal to CPU when data were received or sent. There are 15 registers in each serial port and there are seven registers are re-used. Address lines A0-2 of register logic control were connected with the CPU's bus ADDR0-2 respectively. The Chip Selection (CS) signal nCSA-D of serial ports were independent. The right of Figure 2 is a decoding circuit of low-level CS. Through the CPU-nCS1 Logical OR the ADDR3-6, we can get the base address of 4 serial ports expansion respectively is 0x08000070, 0x08000068, 0x08000058 and 0x08000038 [1,8].

![Serial Extension connection diagram of S3C2410A](image)

Figure 3. Serial Extension connection diagram of S3C2410A

### IV. DRIVER DESIGN

#### A. Initialization of S3C2410A

S3C2410A must be initialized to complete to set the parameter of SC16C554, before the system work [3]. It mainly include: Register the device to the system, Mapped the device address to the virtual address, Set BWSCON, BANKCON Reg to set the timing, Apply for interrupts resources and interrupt mode from system, Set the various expansion serial port communication baud rate, data frame of data bits, stop bits and parity, etc [3, 4, 10].

It was shown at the following initialization function EXT_COM_init(). Note: This system uses the embedded Linux 2.6.11 kernel.

```c
static int __init EXT_COM_init(void) {
    ...
    ret=register_chrdev(EXT_COM_MAJOR,DEVICE_NAME,& EXT_COM_fops); // Register the device to the system
    for(i=0;i<4;i++)
    { vEXT_COM_ADDR[i]=ioremap(pEXT_COM_ADDR[i],16); // Device is mapped to the virtual address
        __raw_writel(0x1f4c,S3C2410_BANKCON1);
        for(i=0;i<4;i++)
        {   
            EXT_COM_Init(vEXT_COM_ADDR[i],EXT_COM_PARAM[i]);
        } //Through the function to set the serial port communications of the baud rate, data frame of data bits, stop bits and parity parameters
        request_irq(EXT_COM_INT[i],uart_irq_handle,SA_INTERRUPT,DEVICE_NAME,NULL);
        // Apply Interrupt Resources
        set_irq_type(EXT_COM_INT[i],IRQT_HIGH);
        // Set the interrupt mode to the high-level trigger
        ...
    return(0);
}
```

#### B. Driver file_operation function description

Several operational functions used in this design are as follows [6]:

- `static int EXT_COM_open(struct inode *inode,struct file *file)` ; // Open device
- `static int EXT_COM_release (struct inode *inode,struct file *file) ` ; // Release resources
- `static ssize_t EXT_COM_read(struct file *file,char *buf,size_t count,loff_t *f_pos) ` ; //Read device
- `static ssize_t EXT_COM_write (struct file,const char *buf,size_t count,loff_t *f_pos) ` ; //Write device
- `static int EXT_COM_ioctl(struct inode *inode, struct file *file,unsigned int cmd,unsigned long arg)` ; //set the serial communication of baud rate, data bits, stop bits and parity parameters
- `static irqreturn_t EXT_COM_irq_handle(int irq,void *dev_id, struct pt_regs *regs); ` // Interrupt handler

Due to space limited, the upper functions are not detail.

### V. EXPERIMENT RESULT

This system uses the embedded Linux 2.6.11 kernel. Test device driver process steps: (1) Load the module, (2) Build the device nodes, (3) Run the program for testing [9].

Run “#insmod sc16c554.ko” to load the module and run “#rmmod sc16c554.ko” to unload the module. Run “#mknod /dev/EXTCOM0 c 233 0 ” to add the first serial port node of expansion. “mknod” command is use to add a node. The first serial port node named EXTCOM0. ‘C’ represents a character device, and ‘B’ represents a block device. The sc16c554 is a character device, so here use ‘C’. The ‘233’ represent the number of major device, and ‘0’ represent the number of minor device [11].

The following is a test application:

In this design, communication parameters of expansion serial ports can be edited, such as baud rate, data bits, stop bits, parity efficacy, flow control, etc. The following is used 9600 bit/s baud rate for the test: (The test program set the communication parameters: Baud Rate: 9600 bit/s, data bits: 8 bit, Stop bit: 1 bit, parity checksum: none, Flow Control: None [13]).
Run ".test" to test the driver, the test is a program file. The running results were shown in Figure 4 and Figure 5.

VI. CONCLUSIONS

In this paper, expansion chip of SC16C554 uses system bus to extend out 4 serial ports which can work independently. SC16C554 uses interrupt work mode to improve the efficiency of ARM9 system, solving the problem of inadequate serial port of embedded systems. If you need to add more expansion serial ports than four, you need one additional SC16C554 chip. Two SC16C554 chips can be expanded to eight serial ports. Using this method, the system can work steadily. Experiments proved to be a viable method of multi-serial ports expansion of embedded system.

REFERENCES

[5] Song Guomin, Study and Implementation of Cost-Effective Multi-Serial Port Expansion, Chengdu Electromechanical College, 2005