TQ2440 Development Platform Manual

Directory

Directory ... 1
Chapter 1 Introduction ... 7
1. 1 Appearance of TQ2440 development platform ... 7
1. 2 Hardware resource of TQ2440 ... 7
1. 3 Software introduction of TQ2440 platform ... 10
1. 4 Introduction of TQ2440 appendix CD-ROM ... 10
Chapter 2 Hardware and Software Environment of Platform ... 14
2. 1 Hardware structure ... 14
2. 1. 1 Hardware resources allocation ... 14
2. 1. 2 Instruction of jumper ... 15
2. 1. 3 Instruction of TQ2440 development platform interface ... 17
2. 2 TQ2440 schematics ... 18
2. 2. 1 power-supply circuit ... 18
2. 2. 2 System reset circuit ... 19
2. 2. 3 System clock circuit ... 19
2. 2. 4 JTAG interface circuit ... 20
2. 2. 5 Serial port circuit ... 20
2. 2. 6 key-press circuit ... 20
2. 2. 7 Buzzer PWM driver circuit ... 21
2. 2. 8 IIC circuit ... 21
2. 2. 9 SD card interface circuit ... 21
2. 2. 10 Real-time clock standby battery power circuit ... 22
2. 2. 11 USB interface circuit ... 22
2. 2. 12 IIS audio data interface circuit ... 23
2. 2. 13 ADC circuit ... 24
2. 2. 14 Ethernet interface circuit ... 24
2. 2. 15 System bus interface ... 25
2. 2. 16 Camera interface ... 26
2. 2. 17 LCD/STN interface circuit ... 26
2. 2. 18 Instruction of Core Board interface ... 27
2. 3 Software characteristics ... 28
2. 3. 1 Linux characteristics ... 28
2. 3. 2 WinCE characteristics ... 29
2. 3. 3 U-Boot characteristics ... 30
2. 4 Windows environment configuration ... 30
2. 4. 1 Hyper-terminal configuration ... 30
2. 4. 2 DNW software configuration ... 34
2. 4. 3 GIVEIO driver intallation ... 35
2. 4. 4 USB download-driver installation ... 44
2. 5 Linux environment configuration ... 52
2. 5. 1 Build cross-compile environment ... 52
2. 5. 2 Network File Service (NFS) configuration ... 55

2. 5. 3 PC Linux FTP service configuration ... 59
2. 5. 4 PC Linux Telnet service configuration ... 60
2. 6 WinCE environment configuration ... 61
Chapter 3 Platform Utilization ... 68
3. 1 Introduction of wire connection on platform and PC ... 68
3. 2 Burning u-boot by SJF2440 ... 68
3. 2. 1 Executing SJF2440 with batch-processing files ... 69
3. 2. 2 Running SJF2440 ... 69
3. 2. 3 Burning u-boot to Nor Flash ... 71
3. 2. 4 Burning u-boot to Nand Flash ... 72
3. 3 Burning u-boot with H-Jtag ... 75
3. 3. 1 H-JTAG installation ... 75
3. 3. 2 H-JTAG configuration ... 75
3. 3. 3 H-Flasher configuration ... 78
3. 4 u-boot application ... 85
3. 4. 1 Function introduction ... 85
3. 4. 2 burning u-boot ... 85
3. 4. 3 Configuration and estimation of u-boot start-up state ... 85
3. 4. 4 Introduction of u-boot utilization ... 86
3. 5 Burning system ... 137
3. 5. 1 Burning Linux OS ... 137
3. 5. 2 Burning WinCE OS ... 137
3. 5. 3 Burning TQ2440-Test_xxxx.bin file ... 137
3. 6 Linux experiment ... 138
3. 6. 1 Experiment of program termination ... 138
3. 6. 2 Experiment of program auto-run configuration ... 138
3. 6. 3 Experiment of setting and saving system real-time clock ... 139
3. 6. 4 Experiment of USB mobile storage device test ... 139
3. 6. 5 Experiment of SD card test ... 140
3. 6. 6 Experiment of mounting NFS ... 141
3. 6. 7 Experiment of USB camera capturing screen test ... 142
3. 6. 8 Experiment of sound card test ... 144
3. 6. 9 Experiment of files transmission with PC via serial port ... 145
3. 6. 10 Experiment of screen capture ... 150
3. 6. 11 Experiment of user LED test ... 151
3. 6. 12 Experiment of user keyboard test ... 154
3. 6. 13 Experiment of log in BBS in telnet ... 155
3. 6. 14 Experiment of remote login platform by telnet ... 156
3. 6. 15 Experiment of FTP remote file transmission ... 157
3. 6. 16 Experiment of Web server test ... 159
3. 6. 17 Experiment of USB camera remote control ... 160
3. 6. 18 Experiment of user LED remote control ... 162
3. 7 WinCE experiment ... 163

3. 7. 1 Experiment of USB mobile storage device test 163
3. 7. 2 Experiment of SD card test 164
3. 7. 3 Experiment of Flash power failure protection 165
3. 7. 4 Experiment of player utilization 166
3. 7. 5 Experiment of 100M network card test 169
3. 7. 6 Experiment of telnet remote log-in 171
3. 7. 7 Experiment of FTP remote file transmission 172
3. 7. 8 Experiment of Web server test 174
3. 7. 9 Experiment of touch-screen correction 174
3. 7. 10 Experiment of USB camera test 178
3. 7. 11 Experiment of image rotation in LCD 181
3. 7. 12 Experiment of 3 serial ports test 182
3. 7. 13 Experiment of recording test 185
3. 7. 14 Experiment of surfing by IE explorer 188
3. 7. 15 Experiment of USB synchronization by ActiveSync 190
3. 7. 16 Experiment of WinCE self-carried game test 190
3. 7. 17 Experiment of Zuma game test 191
3. 8 Experiment of non-OS testing Demo 193
3. 8. 1 Experiment of PWM function test 194
3. 8. 2 Experiment of real-time clock test 195
3. 8. 3 Experiment of ADC conversion test 195
3. 8. 4 Experiment of external interrupt test 196
3. 8. 5 Experiment of touch-screen test 196
3. 8. 6 Experiment of LCD test 197
3. 8. 7 Experiment of IIC interface test 197
3. 8. 8 Experiment of audio output test 198
3. 8. 9 Experiment of audio input test 199
3. 8. 10 Experiment of SD card test 199
3. 8. 11 Experiment of camera test 200
Chapter 4 Linux Development Manual 202
4. 1 Compiling bootloader 202
4. 2 Compiling Linux-2. 6 kernel 205
4. 2. 1 Use configuration file of EmbedSky to compile Linux kernel 205
4. 2. 2 Customizing Linux kernel 210
4. 3 Making root file system 240
4. 3. 1 Components of root file system 240
4. 3. 2 BusyBox compiling 240
4. 3. 3 Building root file system 245
4. 3. 4 Making Yaffs root file system image 246
4.4 Compiling u-boot 247
4.4.1 Decompressing u-boot 247
4.4.2 Configuring u-boot 247
3.4.3 Compiling u-boot 248

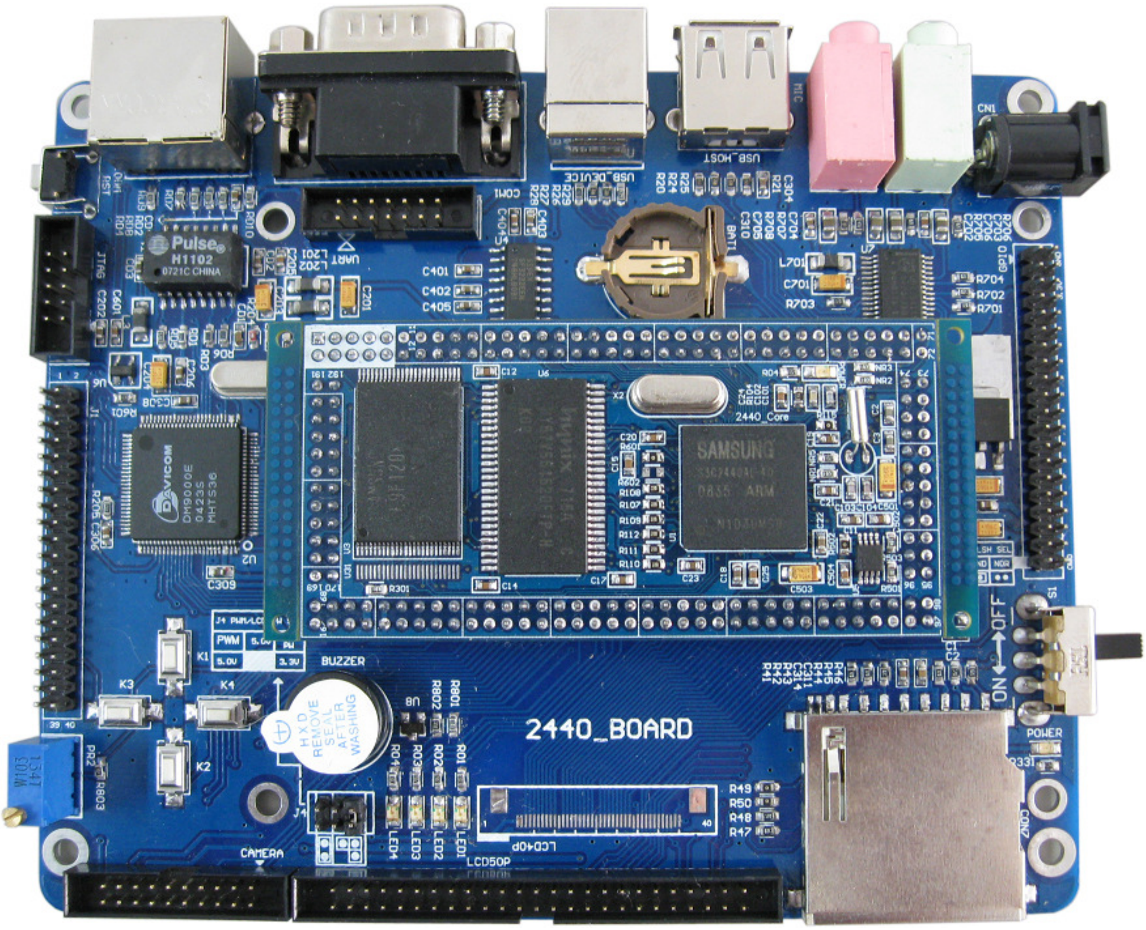
Chapter 5 WinCE development manual........ 250
5. 1 Installing BSP of TQ2440........ 250
5. 2 Compiling the example projects in CD-ROM........ 253
5. 3 Customizing user project files........ 257
5. 4 Export SDK........ 271
5. 4. 1 Configure SDK........ 271
5. 4. 2 Compiling SDK........ 275
5. 4. 3 Finishing the compiling........ 277
5. 5 Communicating with PC synchronously by using ActiveSync........ 277
5. 5. 1 Installing ActiveSync........ 277
5. 5. 2 Installing synchronizing communication USB driver........ 281
5. 5. 3 Utilize ActiveSync synchronizing software........ 285
5. 5. 4 Transferring file with ActiveSync........ 288
5. 6 Capturing screen with ActiveSync and Platform Builder connection........ 290
5. 6. 1 Configuring platform........ 290
5. 6. 2 Configuring PC........ 290
5. 7 Check the platform register information based on connection between ActiveSync and Platform Builder........ 298
Chapter 6 Qt/Embeded graphics development........ 302
6. 1 Simulating Qt/Embeded in PC........ 303
6. 1. 1 Configuring the running evironment........ 303
5. 1. 2 Installation and compiling........ 303
5. 1. 3 Enable the created library........ 304
5. 1. 4 Simulating Qtopia in PC........ 304
6. 2 Programming........ 307
6. 2. 1 Hello routine........ 307
6. 2. 2 Configuring environment parameters........ 308
6. 2. 3 Making desktop starter file........ 308
6. 2. 4 Making the icon........ 308
6. 2. 5 Copying Hello.desktop........ 309
6. 2. 6 Running hello solely........ 309
6. 2. 7 Running hello in Qtopia........ 310
6. 3 Transplanting Web explorer........ 311
6. 4 Building development environment based on ARM........ 317
6. 4. 1 Installing development environment........ 317
6. 4. 2 Make running script........ 317
6. 4. 3 Compiling hello for ARM........ 318
6. 4. 4 Installing hello........ 319
6. 4. 5 Running hello solely on the platform........ 319
6. 4. 6 Running hello in Qtopia on the platform........ 320
6. 4. 7 Make desktop starter file........ 320
6. 4. 8 Make file system containing Qtopia........ 320
Chapter 7 Experiment of driver development........ 328

7. 1 Application development in Linux 328
7. 1. 1 Hello EmbedSky experiment 329
7. 1. 2 Experiment of calling math function 332
7. 1. 3 Experiment of thread programming 333
7. 1. 4 Experiment of UDP network programming 334
7. 1. 5 Experiment of controlling LED 338
7. 1. 6 Experiment of user button controlling 339
7. 2 Example of driver development in Linux 341
7. 2. 1 Hello EmbedSky experiment 342
7. 2. 2 Experiment of LED driver 345
7. 2. 3 Experiment of user keyboard driver 348
7. 3 Experiment of Non-OS application 353
7.3.1 Configuring the experiment environment 353
7. 3. 2 Experiment of test 357
7. 3. 3 Burning TQ2440-Test.bin file 366
Chapter 8 Transplantation of Web server 367
8. 1 Transplanting boa software 367
8. 1. 1 Building compiling environment 367
8. 1. 2 Configuring the compiling condition 367
8. 1. 3 Compiling and optimizing 367
8. 2 Transplanting cgic library 368
8. 2. 1 Building compile environment 368
8. 2. 2 Configuring compile condition 368
8. 2. 3 Compiling and optimizing 369
8. 3 Configuring Web server 369
8. 3. 1 Configuring boa 369
8. 4 Testing 372
8. 4. 1 Static web page test 372
8. 4. 2 CGI script test 372
8. 5 Solving the error 373
8. 5. 1 Error 1 373
8. 6 Some source codes 373
8. 6. 1 cgictest.cpp source code 373
8. 6. 2 Source code of home page 378
8. 7 Environment variable 380
8. 7. 1 The environment variable relevant to the server 380
8. 7. 2 The environment variable relevant to the customer 380
8. 7. 3 The environment variable relevant to the request 380
Chapter 9 Embeded database transplantation (SQLite) 382
9. 1 Introduction of SQLite database 382
9. 2 Transplanting SQLite database 383
9. 2. 1 Get SQLite source code 383
9. 2. 2 Transplanting SQLite 383

9. 2. 3 Using SQLite 385
9. 3 Testing SQLite database 385

Chapter 1 Introduction

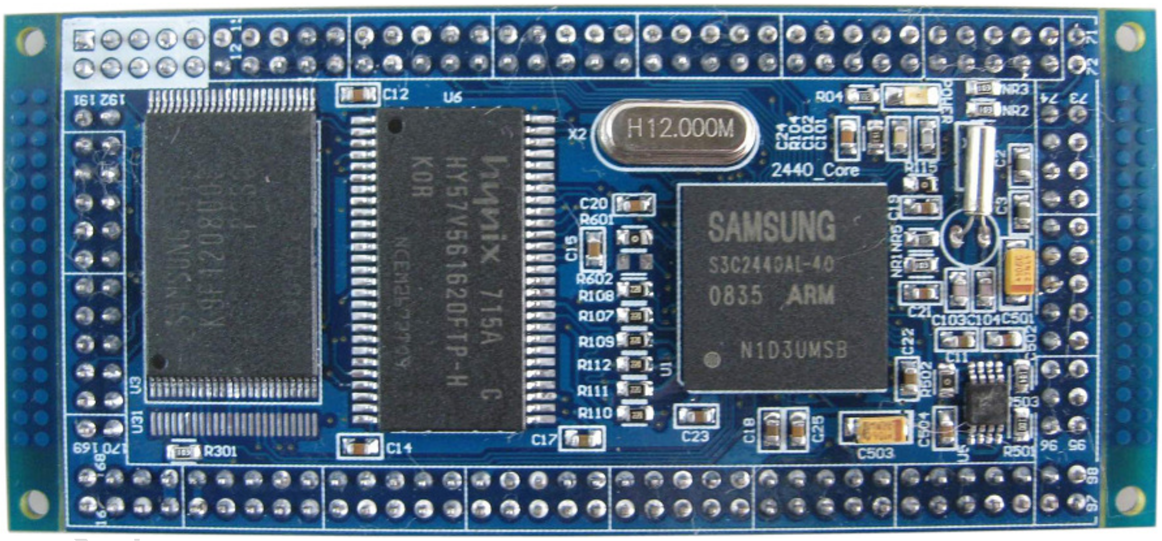
1. 1 Appearance of TQ2440 development platform



1. 2 Hardware resource of TQ2440

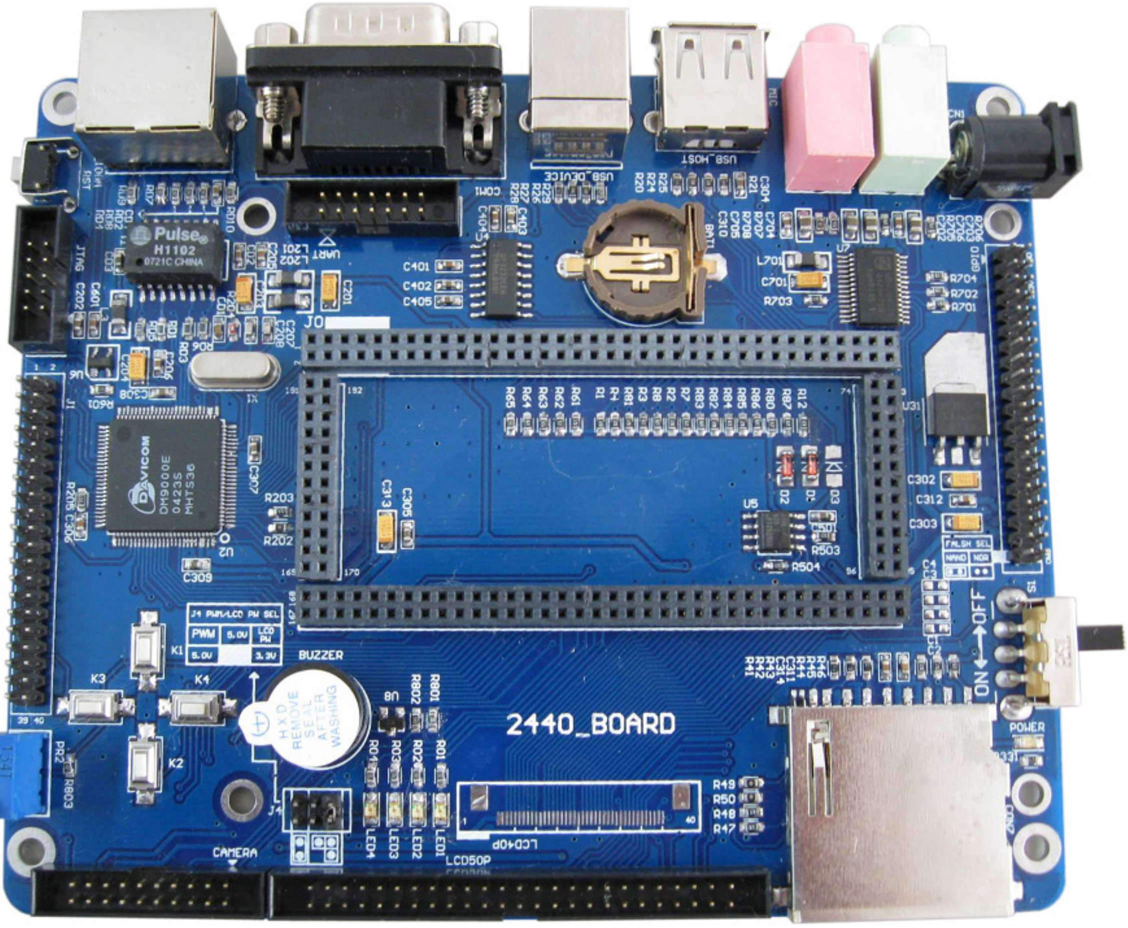
TQ2440 development platform is composed of a Core Board and Mother Board, to make secondary development easier, The CD-ROM contains reference PCB files.

TQ2440 Core Board specifications



- CPU
  - Samsung S3C2440AL，400MHz，highest frequency: 533MHz。
- SDRAM
  - On-board 64MB SDRAM
  - 32bit data bus
  - SDRAM clock frequency as high as 100MHz
- Flash ROM
  - On-board 256MB Nand Flash，nonvolatile
  - On-board 2MB Nor Flash
- 1.25V main power supply ，fully solving the CUP heating problem
- A power indicator light
- 3.3V Core Board power supply

TQ2440 Mother Board specification



◆ Provide 3.3V power supply, load current up to 1.5A, excellent power supply can avoid issues resulting from power instability；

◆ LCD interface

－ On-board LCD interface contains a 4-line resistance type touch-screen interface which can connect to a 4-line resistance type touch-screen directly.

－ Supporting STN LCD of monochrome, 4-level gray, 16-level gray, 256-level gray and 4096-level color, sized from 3.5 inch to 12.1 inch, resolution up to 1024×768 pixels.

－ Supporting TFT LCD of monochrome, 4-level gray, 16-level gray, 256-level gray and 64K-level color and true color, sized from 3.5 inch to 12.1 inch, resolution up to 1024×768 pixels.

－ Standard TQ2440 suite contains TFT true color TFT LCD of 256K-level color, 3.5 inch dimension and 240×320 pixels resolution. And a touch-screen.

◆ A 100M Ethernet RJ-45 interface.

◆ 3 serial port interfaces. COM1 is a DB9 interface with voltage converted by RS3232 (Including interface of CMOS voltage level). The UART interface on-board is namely the extended interface of serial port with CMOS voltage level.

◆ A USB type Host A interface (supporting USB1.1 protocol).

◆ A USB type Slave B interface (supporting USB1.1 protocol).

◆ A SD memory card interface, supporting DMA transmission mode.

◆ 1-way stereo audio output interface, 1-way audio input interface.

◆ A 2.0mm, 10-Pin Jtag interface, which can be used for software simulation and step debugging and download u-boot.

- 4 User Buttons.
- 4 on-board user LED lights.
- On-board AD test unit
- On-board PWM function test unit (buzzer).
- On-board EEPROM test unit.
- A 20-Pin, 130 megapixels CMOS camera interface.
- A 40-Pin GPIO extended interface.
- A 40-Pin bus extended interface.
- On-board real-time clock battery.
- Power switch and indicator light

Specification size

- 78mm×37mm (Core Board)
- 128mm×105mm (Mother Board)

1. 3 Software introduction of TQ2440 platform

Operating system provided

- Window CE 5.0
- Linux-2.6.13
- uCOS-II

Bootloader provided

- u-boot-1.1.6 (supporting TFTP transmission and burning yaffs file system)

Test program provided

- Non-OS test program

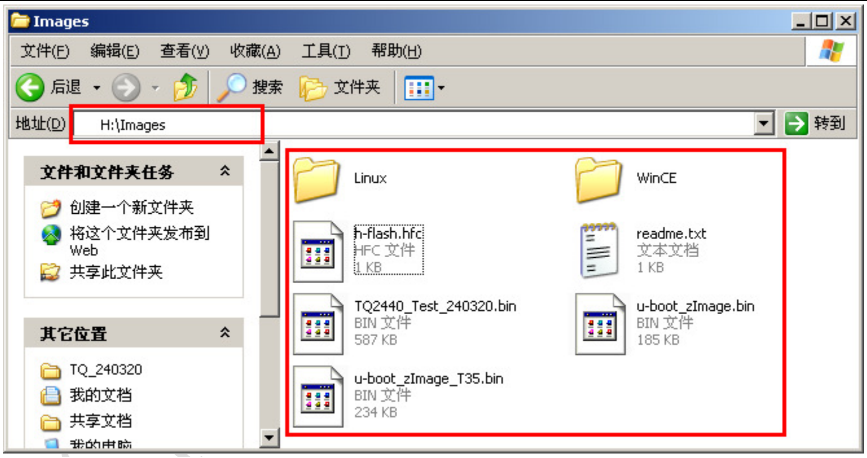
Schematics provided

- Schematics of Core Board (PDF format).
- Schematics and PCB diagram of Mother Board (original diagram for board manufacturing).

1. 4 Introduction of TQ2440 appendix CD-ROM

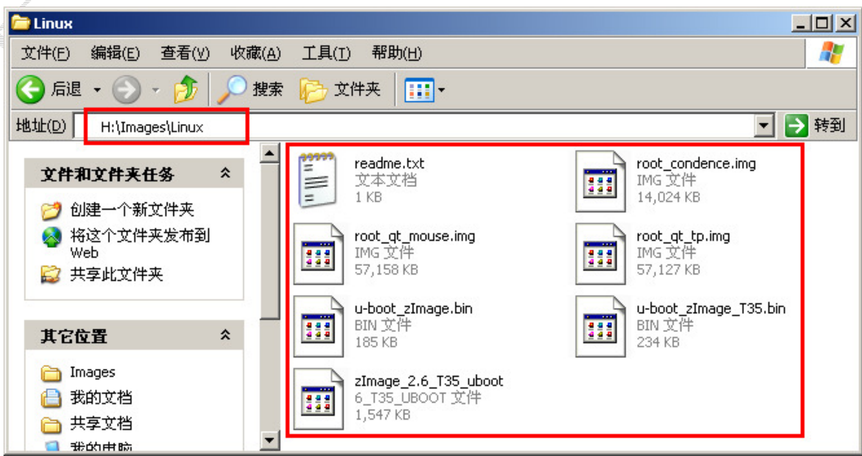
- Images directory

The directory contains compiled image files, including uboot image, non-os test program image, linux image and WinCE image (under the directory WinCE)



→ Linux directory

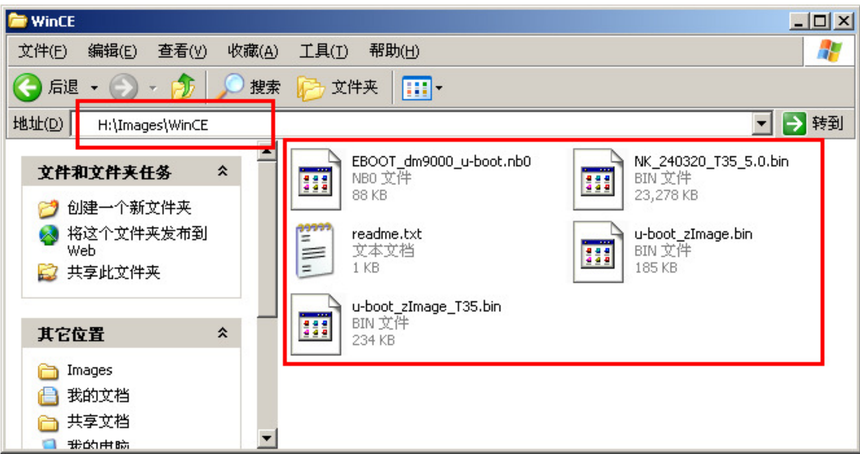
The compiled Linux image files, including: kernel image, simplified file system image, Qt image supporting touch-screen version and Qt image of mouse version.



→ WinCE directory

Compiled WinCE image file, including: eboot image and NK image.

Caution: the resource code of uboot and eboot image are not contained in CD-ROM; If there is a need of eboot resource code, please contact us.



➢ Linux directory

Containing: linux kernel, file system, Qt/Embeded, Busybox, uboot, database, web server, cross-compiler and example program.

Caution: The source code of uboot contained in CD-ROM does not support USB download mode, but only TFTP download mode.

- WinCE directory

Contains: BSP package, project files and SDK package.

→ SMDK2440 directory

WinCE 5.0 BSP package

→ TQ2440 directory

Project file sample, the file NK.bin under the directory Images\WinCE\ is compiled from it.

→ SDK directory

The SDK package exported from project file sample.

- Windows platform tools

Containing general softwares and drivers.

→ ActiveSync

WinCE synchronizing software, needs to be intalled to activate WinCE synchronization function.

→ DNW

Running in PC server, used for transmitting data to platform in USB download function.

→ GIVEIO

A driver used to virtualize the parallel port into normal IO driver when using Jtag.

→ H-JTAG

Used for software simulation and burning uboot into Nor Flash.

→ Jave patch

This patch is indispensable when accessing USB camera via web explorer.

→ SJF2440

A program running in PC for Jtag download.

→ TFTP_Server_TFTPDWIN_v0.4.2

A proxy software runnig in PC for TFTP download.

→ USB driver

The driver used for USB download function.

→ VMware

A software used for installing Linux in Windows

→ WinCE synchronization driver

A USB driver used for synchronization between WinCE and PC

- Circuit diagram

Containing Core Board schematic, Mother Board schematic and Mother Board PCB diagram.

- Information from Samsung website

Containing some resources downloaded from Samsung company website.

- Non-OS test program

The test program running independent from OS.

- Chip manual

Containing information of all chips appearing in development platform.

- User manual

Containing development information of TQ2440

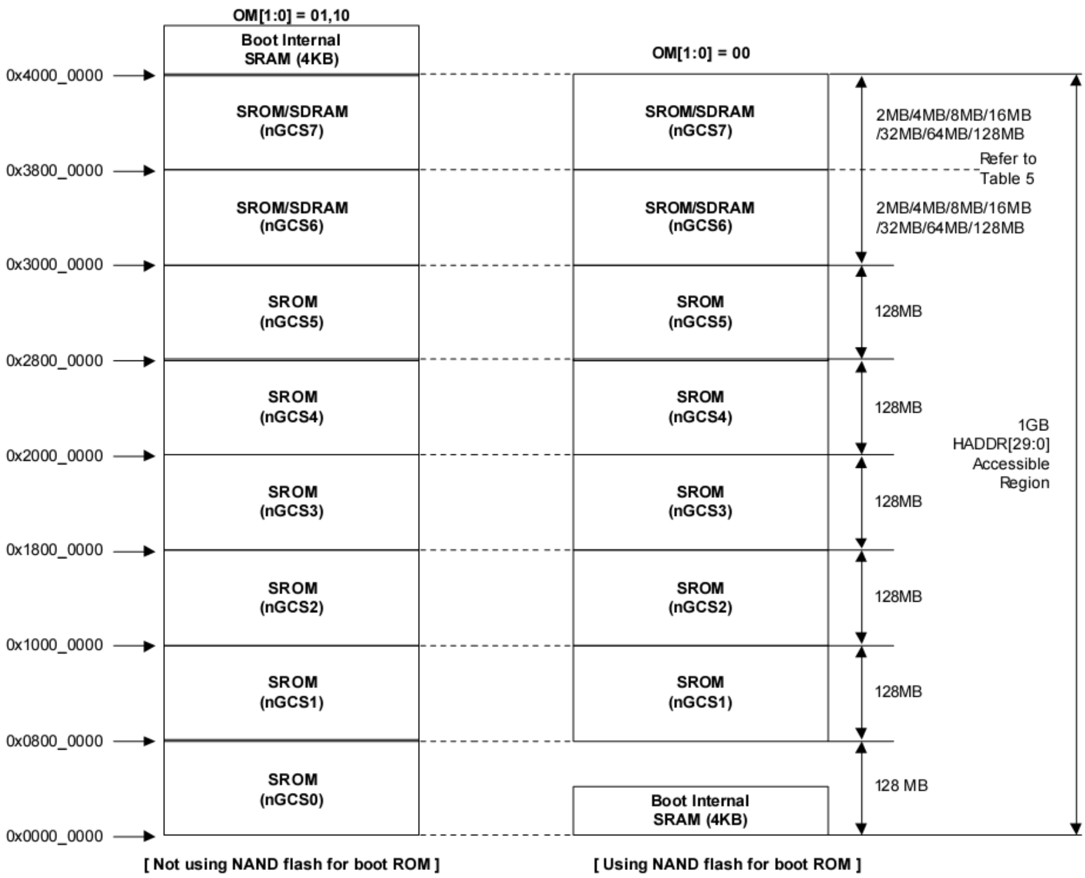
Chapter 2　Hardware and Software Environment of Development Board

2. 1 Hardware structure

2. 1. 1 Hardware resources allocation

Address space allocation and chip selection signal definition

S3C2440 supports 2 kinds of start-up modes: Start up from Nand Flash, and start up from Nor Flash. The address allocations of chip selection are dfferent in these 2 modes, as shown in the following diagram:



The upper chart on the left indicates the address allocation in Nor Flash start-up mode of nGCS0; The upper chart on the right describepts the address allocation in Nand Flash start-up mode.

(Note: Use the jumper OM0 in Mother Board to select Nand Flash and Nor Flash start-up mode. Nand Flash start-up mode is selected when the jumper is fit on, and Nor Flash start-up mode is selected when the jumper is removed.)

In the upper graph, the Nand Flash start-up mode, 4KB Boot Internal SRAM in CPU is mapped into the chip-selection space of nGCS0. Before a program starts, CPU copies the first 4KB part of the proram into this space. Then the program starts to run. If a program is larger than 4KB, the first part of 4KB needs to include the code initializing Nand Flash and other devices, and copyingies data from Nand Flash into SDRAM on-board, and jumping to SDRAM space on-board from the 4KB space;

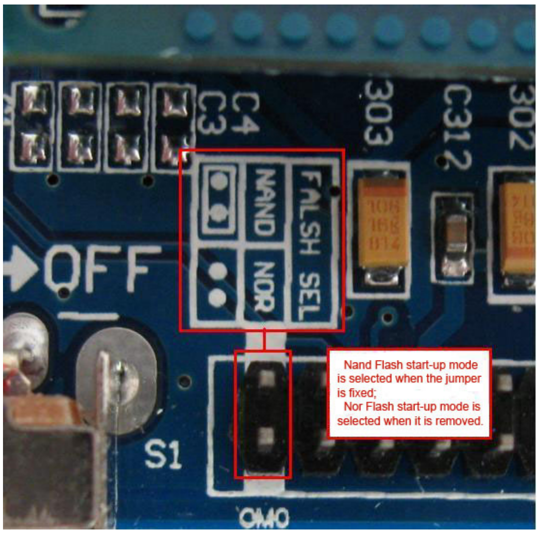
Under the Nor Flash start-up mode, nGCS0 is mapped from the beginning address of Nor Flash.

SDRAM address space on-board: 0x30000000~0x34000000.

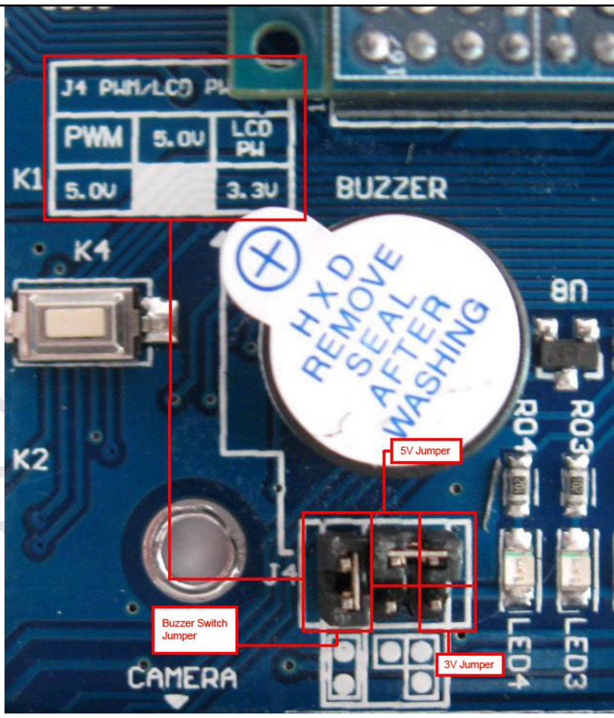
2. 1. 2 Explanations of jumper

There are 2 jumpers in TQ2440 platform (J4 in Mother Board of TQ2440 and OM0 of GPIO interface):

The jumper OM0 of GPIO extended interface refers to Boot Select，the Nand Flash start-up mode is selected when the jumper is fixed; Nor Flash start-up mode is selected when it is removed.

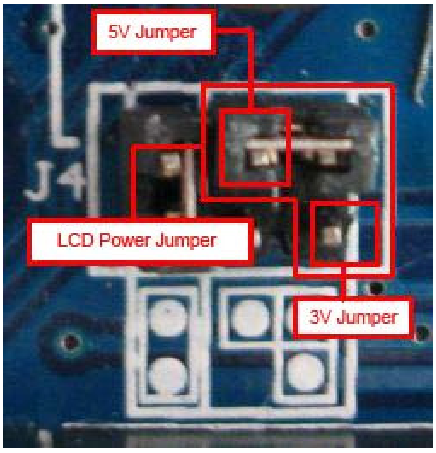


The role of jumper J4: Switch on LCD power, and switch on the buzzer, as shown in the following diagram.

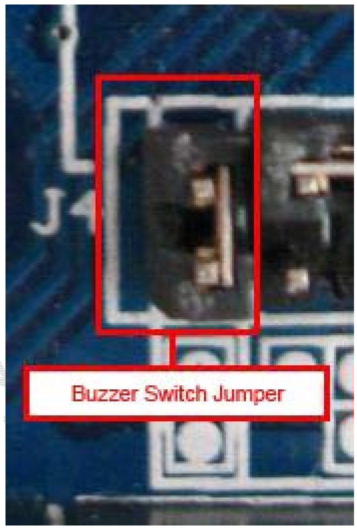


The “L” shaped jumper interface refers to switch on LCD powering, selecting 3.3V or 5V voltage to power LCD.

Samsung 3.5 inch LCD and Donghua 3.5 inch LCD needs a 3.3V power; Toshiba 3.5 inch LCD and Samsung 7 inch LCD needs a 5V power. As shown in the following diagram.

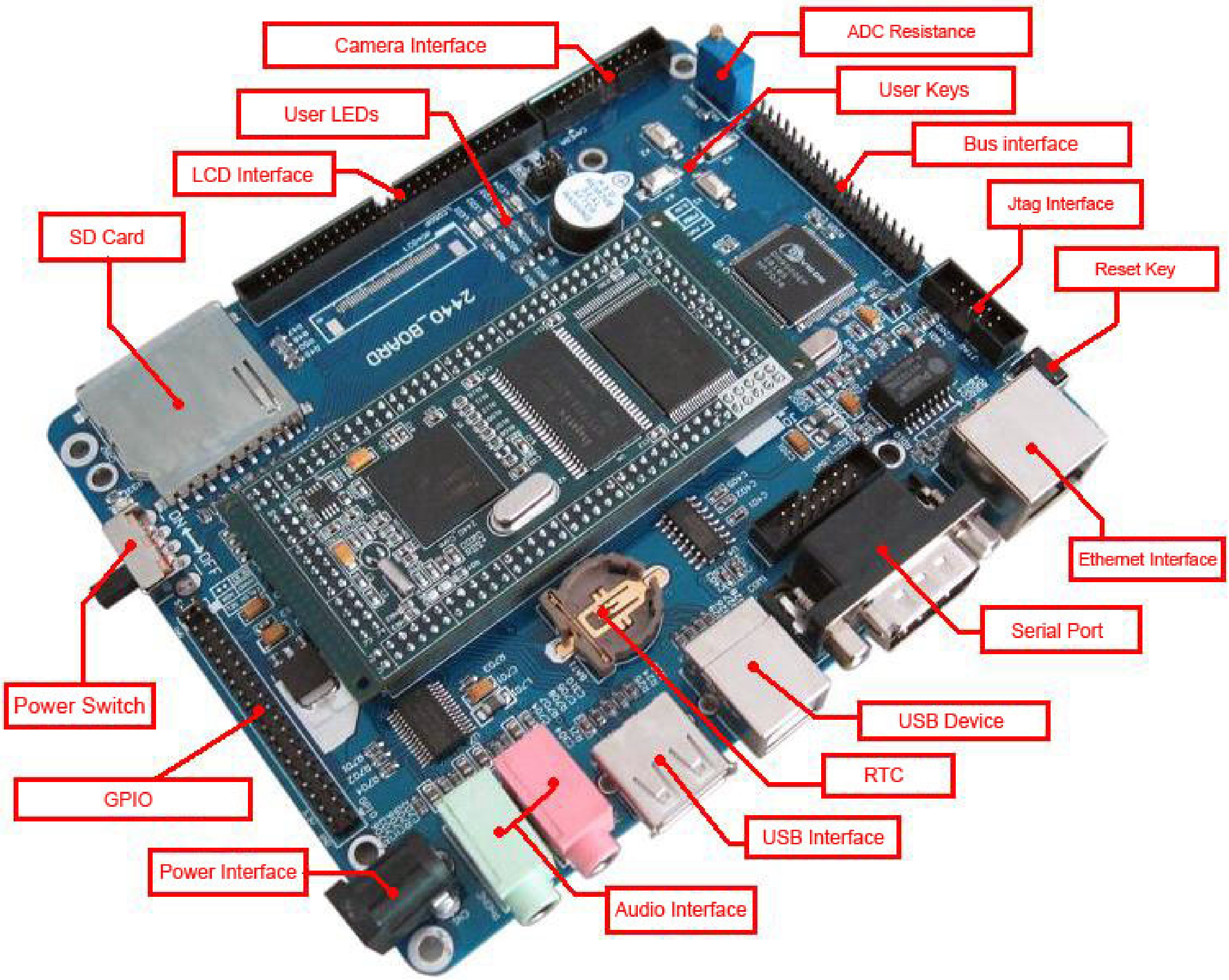


The “I” shaped jumper interface plays a role of power switch of the buzzer. The power is on when the jumper is fixed, and is off when it is removed.



2. 1. 3 Explanations of TQ2440 development platform interface

The main interfaces of TQ2440 is shown in the following picture:



◆ **Power interface**

Caution: the input power needs to be less than 7V. The output voltage of power adapter provided is 5V. Make sure to confirm your own power-adapter output by consulting the supplier or by measuring with a reliable device, in order to prevent from an excessive high input.

Caution: A reliable 5V power-adapter, like the one provided by us is the best choice for the power source.

◆ **Audio interface**

TQ2440 provides standard audio interfaces, the green one is for output, and the red one for input.

◆ **USB interface**

TQ2440 development board provides 2 USB interfaces, one is USB A (USB_HOST interface，be used to connect to the U disk, USB camera and other devices); The other is USB B (USB_Deive interface, connect to PC with a USB extension-line to transmit data).

When using USB for download function, using a standard USB extension-line to connect PC and development platform. Advice: Remove the extension-line immediately after the download is complete.

◆ **Serial port**

Serial port is one of the most important interfaces of TQ2440 development board. It is used for interaction and data transmission between platform and PC, and debugging.

We offer a standard direct-connecting serial port line to connect development platform and PC for interaction.

◆ **Network card interface**

TQ2440 carries a 100M network card interface. The user can connect the platform with a net line and go surffing in operating system; In uboot-download mode, the user could download data to development platform by using TFTP via network card interface.

◆ **Jtag interface**

In TQ2440, if there is no uboot in Nand Flash or in Nor Flash. Using Jtag to burn uboot; Or using Jtag for simulation

Connect Jtag interface and PC parallel port with Jtag line, using Jtag software to burn program or to simulate.

Note: Remove the Jtag line if not using it.

◆ **Camera interface**

When connecting the camera, keep lens outward.

◆ **LCD interface**

Beware of the direction of LCD interface and don’t make it reversed when connecting.

Caution: When using Toshiba LCD, be careful don’t touch the high-voltage region on backside of the LCD driver board, In order to prevent from electronic shock.

◆ **SD card interface**

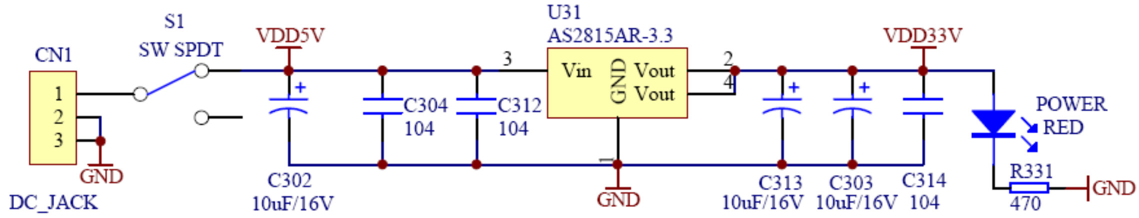
Insert the SD card with its interface side downwards.

2. 2 TQ2440 schematics

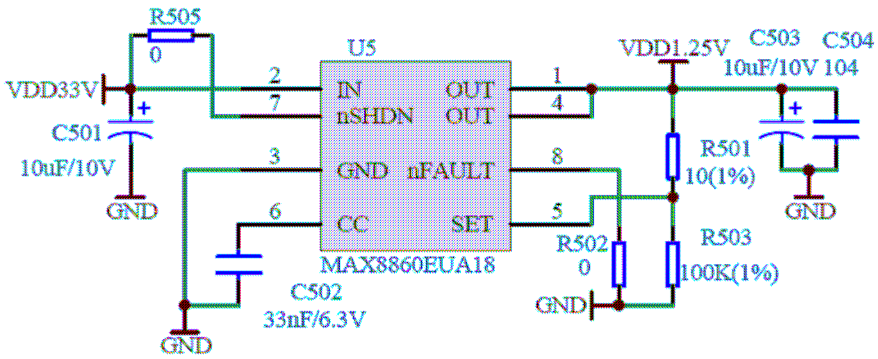
2. 2. 1 power-supply circuit

TQ2440 supports a 5V input. The platform contain power switch and indicator light. A Low Dropout Line Regulator AS28 M76D7V33915AR-3.3 IC which could carry 1.5A load provides platform with 3.3V power supply.

A Low noise and Low Dropout Line Regulator MAX8860EUA provides CPU on Core Board with 1.25V power supply. As shown in the following diagram:



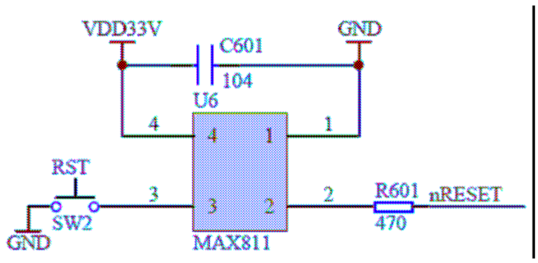
(a) 3.3V power-supply circuit



(b)1.25V power-supply circuit

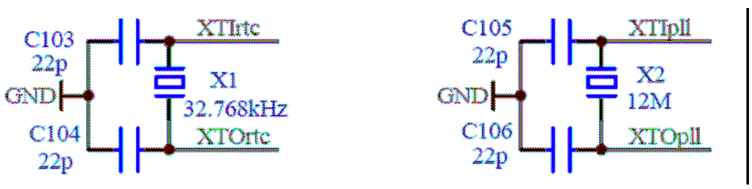
2. 2. 2 System reset circuit

A system reset chip MAX811S is selected to enhance the power supervision ability. If the system voltage is lower than the threshold 2.93V, MAX811S resets the system immediately. As shown in the following diagram:



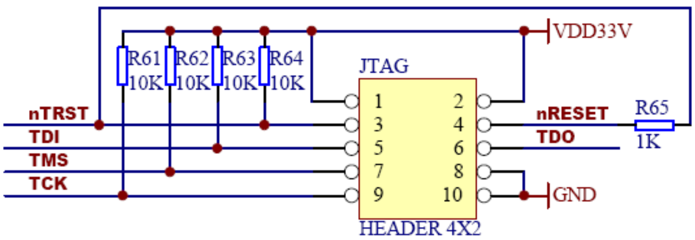
2. 2. 3 System clock circuit

An external oscillator is intended for the source of system clock; The internal PLL circuit could adjust the system clock to speed up the system operation. The system clock circuit is shown in the following diagram:



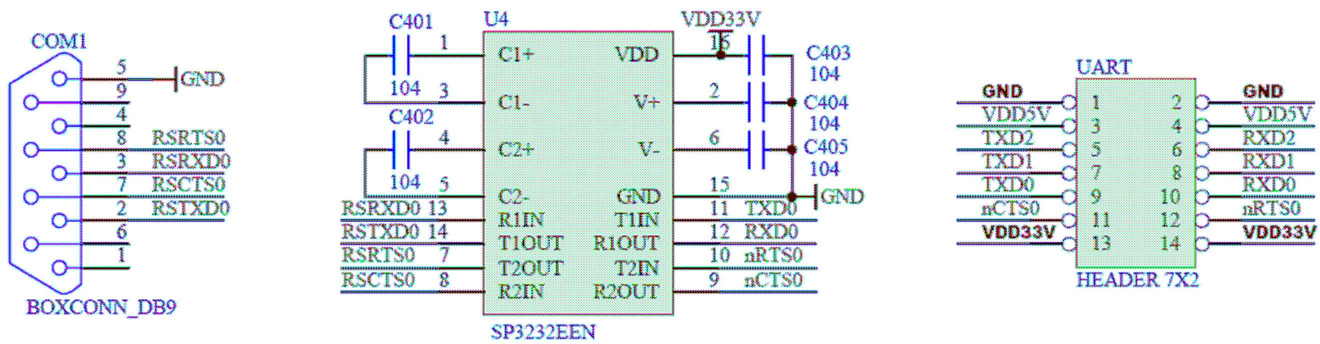
2. 2. 4 JTAG interface circuit

The 10-pin, 2.0mm clearance Jtag interface takes up only a small area on platform. S3C2440 supports JTAG function. The user can use external JTAG debug line or simulator for debugging. The circuit is shown in the following diagram:



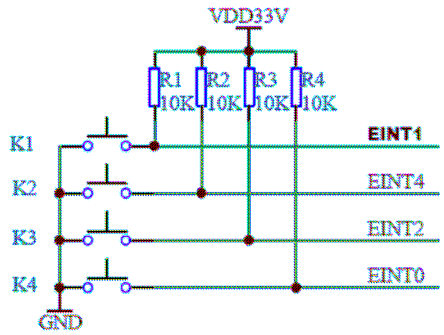
2. 2. 5 Serial port circuit

There is a 5-line asynchronous serial port interface and a UART extension interface. The circuit is shown in the following diagram:



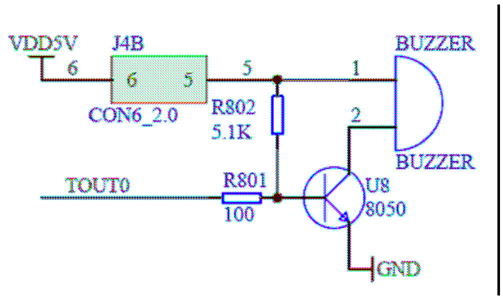
2. 2. 6 key-press circuit

TQ2440 provdes a 4-way key-press circuit, 4 pull-up 10kΩ resistances are connected to 4 GPIO pins on CPU. The voltage level of GPIO turns from high to low when the key is pressed. By using polling program or interrupt the user can be acquainted with the voltage change of GPIO pins. The circuit is shown in the following diagram:



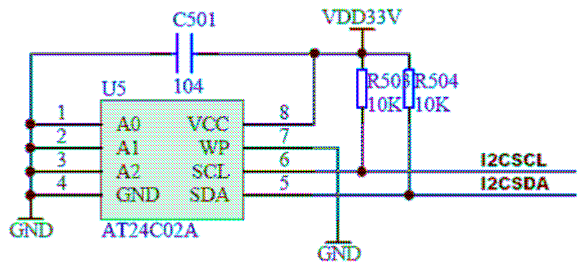
2. 2. 7 Buzzer PWM driver circuit

The buzzer on platform is driven by PWM, and produces sound with different frequencies. The circuit is shown in the following diagram:



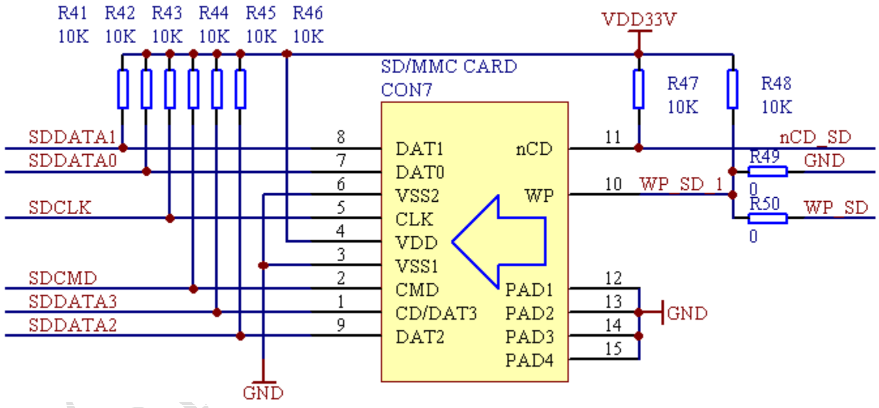
2. 2. 8 IIC circuit

I^2C circuit supports a multi-master-controlling I^2C bus serial interface. A serial data bus SDA and a serial clock bus SCL transmit data between master and slave device. SDA and SCL are both duplex and are used for reading and writing operation of AT24C02A. I^2C circuit is shown in the following diagram:



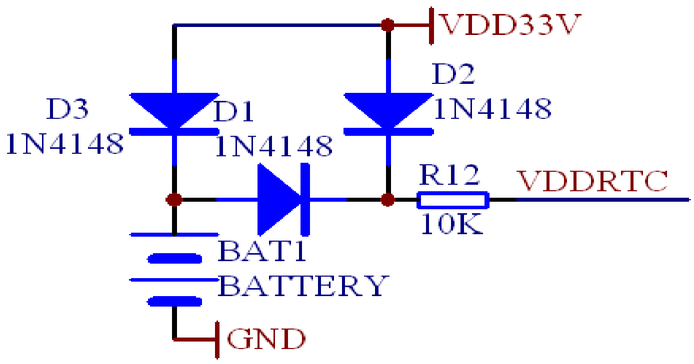
2. 2. 9 SD card interface circuit

SD (Security Digital) card is a kind of widely applied card. A specified interface circuit on platform supports reading and writing function of SD card. S3C2440 integrates SD module in itself. The circuit is shown in the following diagram:



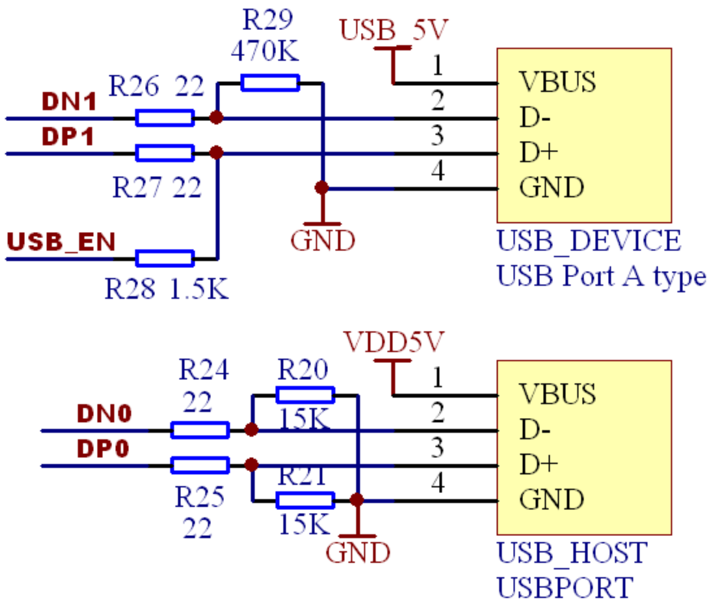
2. 2. 10 Real-time clock standby battery power circuit

The standby battery protects system state data during power-off. The reference circuit is shown in the following diagram:



2. 2. 11 USB interface circuit

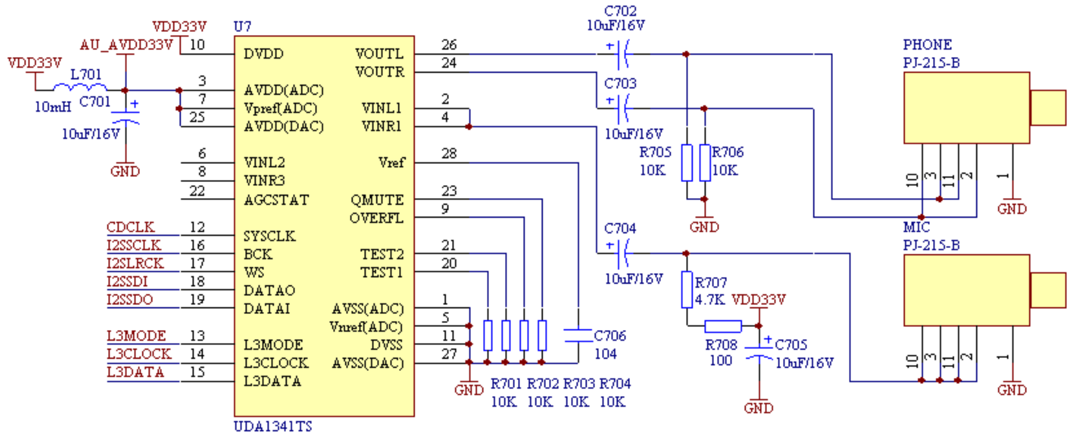
S3C2440A integrates USB module itself, containing a HOST USB1.1 interface and a Device USB1.1 interface. The circuit is shown in the following diagram:



2. 2. 12 IIS audio data interface circuit

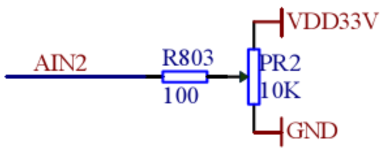
I^2S (Inter-IC sound bus), is a interface standard introduced by Philips and Sony company. The I^2S circuit is shown in the diagram 2-12. The system connects I^2S interface and UDA1341TS (provided by Philips company), to build SPEADER audio ouput channel and MICROPHONE audio input channel. UDA1341TS can convert digital signal into analog signal, and can convert analog stereo signal into digital signal, and can also use AGC (Auto Gain Control), PGA (Programmable Gain Adjust) to process analog signal. The chip supports DSP function in order to process digital signal. In practical application, UDA1341TS can be widely used in CD, MD and digital camera. The bus L3 of UDA1341TS is used when the chip works in micro-controller-input mode. L3 includes L3DATA (data line of interface), L3MODE (mode line of interface) and L3CLOCK (clock line of interface). The processor can configure the audio-processing parameter and system-controlling parameter of UDA1341TS via this interface.

As shown in the following diagram:



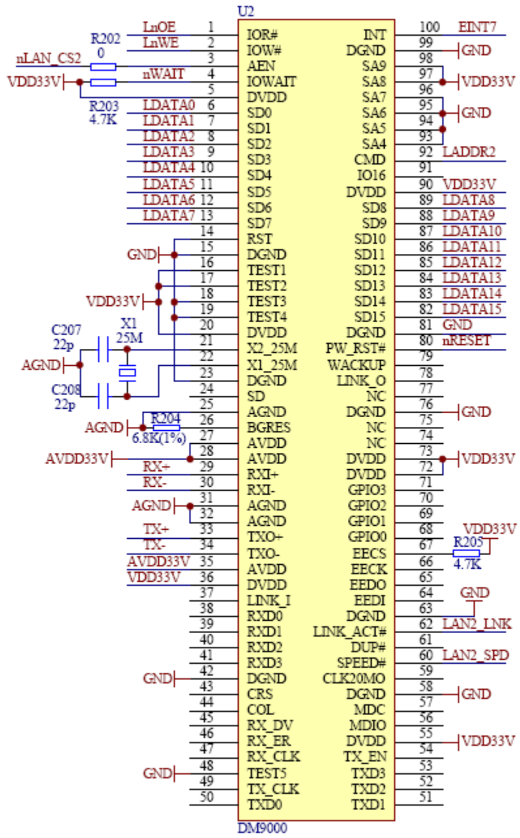
2. 2. 13 ADC circuit

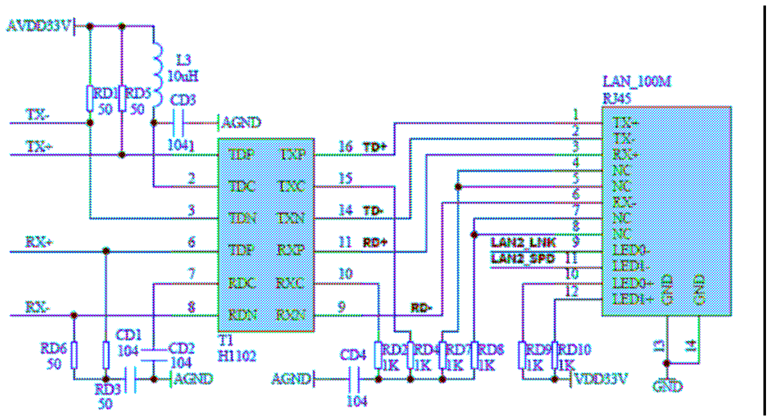
S3C2440A has an 8-way, 10-bit CMOS A/D converter, and 3.3V reference voltage. The platform carries a DC voltage test circuit. The resistance PR2 is used to adjust input voltage. AIN2 is the analog voltage input. As shown in the following diagram:



2. 2. 14 Ethernet interface circuit

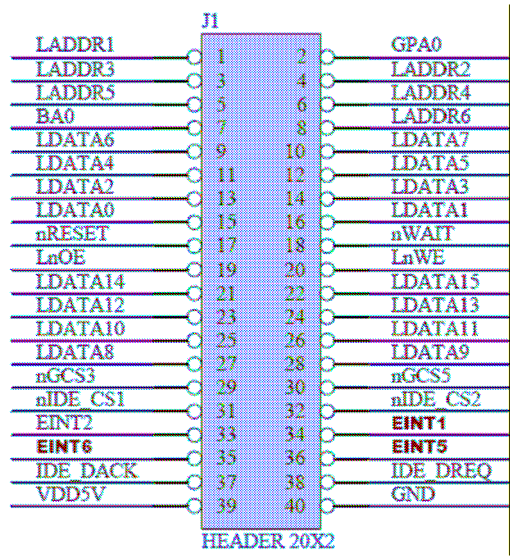
S3C2440A carries no network interface. By adopting extension-network-interface mode, the platform provides a DM9000E 100M network interface. The circuit is shown in the following diagram:





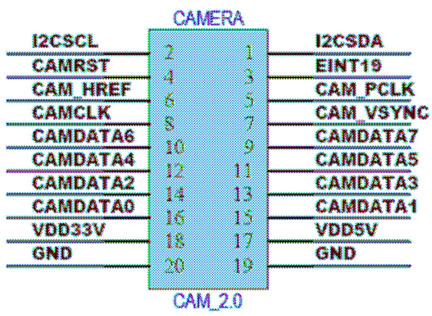
2. 2. 15 System bus interface

System bus interface is made up of 8 address-buses (address 0~6 and address 24), 16 data-buses, 4 interrupt-buses and 4 chip-selection buses. System bus integrates leading points needed by IDE for extension. The circuit is shown in the following diagram:



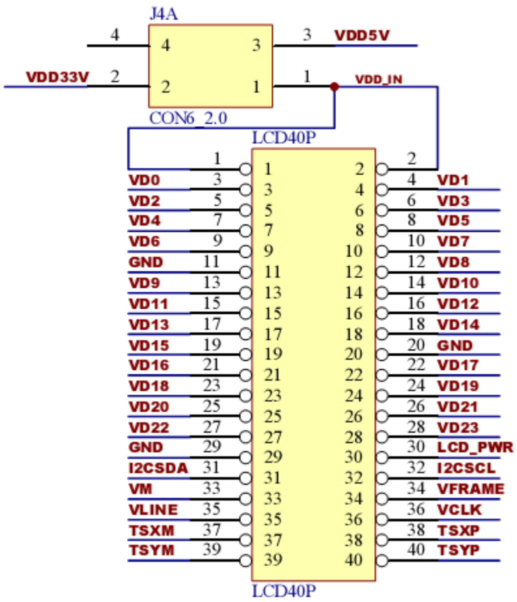
2. 2. 16 Camera interface

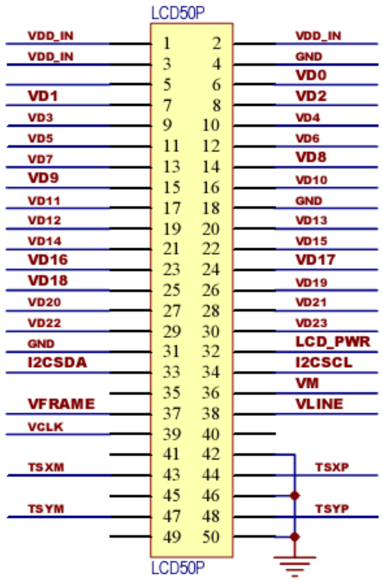
S3C2440A integrates the CAMERA module in itself. The user can connect different types of camera to the interface (like OV9650). The circuit is shown in the following diagram:



2. 2. 17 LCD/STN interface circuit

TQ2440 support 2 types of interface, 50-pin 2.0mm-clearance and 40-pin 0.5mm-clearance, which can support monochromatic, pseudo-color, true-color and touch-screen LCD interface. The circuit is shown in the following diagram:





Caution：Use jumper J4 to select 3.3V or 5V power for LCD.

2. 2. 18 Instruction of Core Board interface

pin	function	pin	function	pin	function	pin	function
1	DATA6	2	DATA7	3	ADDR7	4	ADDR8
5	ADDR5	6	ADDR6	7	ADDR3	8	ADDR4
9	ADDR1	10	ADDR2	11	DATA30	12	DATA31
13	DATA28	14	DATA29	15	DATA26	16	DATA27
17	DATA24	18	DATA25	19	DATA22	20	DATA23
21	DATA20	22	DATA21	23	DATA18	24	DATA19
25	DATA16	26	DATA17	27	nTRST	28	nRESRT
29	TDO	30	TDI	31	TCK	32	TMS
33	RXD2/nCTS1/GPH7	34	TXD2/nRTS1/GPH6	35	RXD1/GPH5	36	TXD1/GPH4
37	RXD0/GPH3	38	TXD0/GPH2	39	nRTS0/GPH1	40	nCTS0/GPH0
41	EINT0/GPF0	42	EINT1/GPF1	43	EINT2/GPF2	44	EINT3/GPF3
45	EINT4/GPF4	46	EINT5/GPF5	47	EINT6/GPF6	48	EINT7/GPF7
49	EINT8/GPG0	50	EINT11/nSS1/GPG3	51	EINT14/SPIMOSI1/GPG6	52	EINT13/SPIMISO1/GPG5
53	EINT19/TCLK1/GPG11	54	EINT15/SPICLK1/GPG7	55	EINT18/nCTS1/GPG10	56	EINT9/GPG1
57	EINT20/GPG12	58	VDD_RTC	59	DP1/PDP0	60	AIN3
61	DN1/PDN0	62	AIN2	63	DN0	64	AIN1
65	DP0	66	AIN0	67	EINT13/SPIMISO1/GPG5	68	EINT10/nSS0/GPG2
69	SPICLK0/GPE13	70	SPIMOSI0/GPE12	71	EINT22/GPG14	72	EINT21/GPG13
73	Vref	74	EINT23/GPG15	75	OM2	76	OM3
77	OM0	78	OM1	79	EINT16/GPG8	80	SDDAT2/GPE9

81	SDDAT3/GPE10	82	SDCMD/GPE6	83	SDCLK/GPE5	84	SDDAT0/GPE7
85	SDDAT1/GPE8	86	UEXTCLK/GPH8	87	TCLK0/GPB4	88	CDCLK/ GPE2
89	I2SLRCK/ GPE0	90	I2SSCLK/ GPE1	91	TOUT3/GPB3	92	TOUT2/GPB2
93	I2SSDI/ GPE3	94	I2SSDO/ GPE4	95	EINT12/LCD_PWREN/GPG4	96	XP/AIN7
97	XM/AIN6	98	YP/AIN5	99	YM/AIN4	100	VCLK/GPC1
101	VLINE/GPC2	102	VFRAME/GPC3	103	VM/GPC4	104	IICSCL/GPE14
105	IICSDA/GPE15	106	VD23/nSS0/GPD15	107	VD22/nSS1/GPD14	108	VD21/ GPD13
109	VD20/ GPD12	110	VD19//GPD11	111	VD18/SPICLK1/GPD10	112	VD17/SPIMOSI1/GPD9
113	VD16/SPIMISO1/GPD8	114	VD15/GPD7	115	VD14/GPD6	116	VD13/ GPD5
117	VD12/GPD4	118	VD11/GPD3	119	VD10/GPD2	120	VD9/GPD1
121	VD8/GPD0	122	VD7/GPC15	123	VD6/GPC14	124	VD5/GPC13
125	VD4/GPC12	126	VD3/GPC11	127	VD2/GPC10	128	VD1/GPC9
129	VD0/GPC8	130	nXDACK0/GPB9	131	nXDREQ0/GPB10	132	nXBACK/GPB5
133	nXBREQ/GPB6	134	nXDACK1/GPB7	135	nXDREQ1/GPB8	136	TOUT1/GPB1
137	TOUT0/GPB0	138	CAMRESET/GPJ12	139	CAMVSYNC/GPJ9	140	CAMHREF/GPJ10
141	CAMPCLK/GPJ8	142	CAMCLKOUT/GPJ11	143	CAMDATA0/GPJ0	144	CAMDATA1/GPJ1
145	CAMDATA2/GPJ2	146	CAMDATA3/GPJ3	147	CAMDATA4/GPJ4	148	CAMDATA5/GPJ5
149	CAMDATA6/GPJ6	150	CAMDATA7/GPJ7	151	nWAIT	152	nGCS1/GPA12
153	nGCS2/GPA13	154	nGCS3/GPA14	155	nGCS4/GPA15	156	nGCS5/GPA16
157	nGCS7	158	nBE1	159	GND	160	GND
161	3.3V	162	3.3V	163	DATA8	164	DATA9
165	DATA10	166	DATA11	167	DATA12	168	DATA13
169	DATA14	170	DATA15	171	ADDR24 /GPA9	172	ADDR0/GPA0
173	nWE	174	nOE	175	ADDR20/GPA5	176	ADDR19/GPA4
177	ADDR18/GPA3	178	ADDR17/GPA2	179	ADDR16/GPA1	180	ADDR15
181	ADDR14	182	ADDR13	183	ADDR12	184	ADDR11
185	ADDR10	186	ADDR9	187	DATA0	188	DATA1
189	DATA2	190	DATA3	191	DATA4	192	DATA5

In the upper definitions, "/" means the multi-function pin. The user can use software to define the pin function; The red letters indicate the default function chosen by platform; The blue letters indicate the audio function chosen by platform; The green letters indicate the network function chosen by platform.

2. 3 Software characteristics

2. 3. 1 Linux characteristics

- Version: 2.6.13
- Support file system
  - Yaffs — readable and writable file system, default selection.
  - Cramfs — compressed read-only file system, not recommended.

- Ext2 — available when connecting a hard-disk
- Fat32 — be used in mobile storing device.
- NFS — network file system, can be used for debugging applications and drivers

◆ The drivers included in CD (Caution: Camera drivers are provided in the form of module, and others are provided in source code.)

- 3 standard serial port drivers.
- DM9000 driver.
- Sound card driver (support playing and recording).
- RTC driver (preserve time record when power-off).
- User LED driver.
- User key-press driver.
- Ordinary LCD drivers (including: resolution 320×240, 240×320 and 640×480).
- Camera driver.
- Touch-screen driver.
- USB camera (including: OV511 camera and others).
- USB mouse, USB keyboard, U disk, mobile hard disk and so on.
- SD card driver.

◆ Linux application and service program.

- busybox1.2.0 tool kit (includes general Linux instructions).
- Telnet、FTP、inetd (remote login tools and services).
- Boa (Web server application).
- Madplay (MP3 player in console).
- Servfox (the camera application in console is able to access USB camera via network and display the image captured by camera in LCD).
- Spcacat (the shooting application of camera in console).
- rz and sz (file receiving and sending application via serial port in console).
- Snapshot (image-capture software in console).
- camera_test (test program of camera in console).

◆ Graphic interface (the source code is provided).

- Qt/Embedded

2. 3. 2 WinCE characteristics

Windows CE 5. 0

◆ Drivers in CD.

- DM9000 network card driver.
- USB wireless lan driver (VNUWLC41).
- Camera driver.
- USB mouse, USB keyboard, U disk and mobile hard disk driver.
- USB camera driver.
- USB synchronization driver.

- 3 serial ports driver.
- Sound card driver.
- SD card driver.
- real-time clock driver.
- register table preservation.
- Flash free space power-off preservation, about 30M free space.

◆ System characteristic (simplified Chinese system)

- Windows XP interface style.
- Windows Media Player 9.0 (supports MP3, MPEG2, MPEG4, WMV, WAV and so on).
- Picture explorer, wordpad.
- IE6 explorer.

2. 3. 3 U-Boot characteristics

The uboot of this platform support both USB download and WinCE download function. The other features supported are list in the following:

- Support Nand Flash and Nor Flash self-adaptive start up.
- Support USB data transmission. The transmission rate reaches as high as 850KB/S.
- Support TFTP network data transmission.
- Support start-up logo.
- Support WinCE and Linux start-up scroll bar.
- Support writing Yaffs file system.
- Support analysis and writing NK.bin file.
- Support WinCE and Linux self-adaptive start up.
- Support writing user program to Nand Flash.
- Support download user program to SDRAM.

2. 4 Windows environment configuration

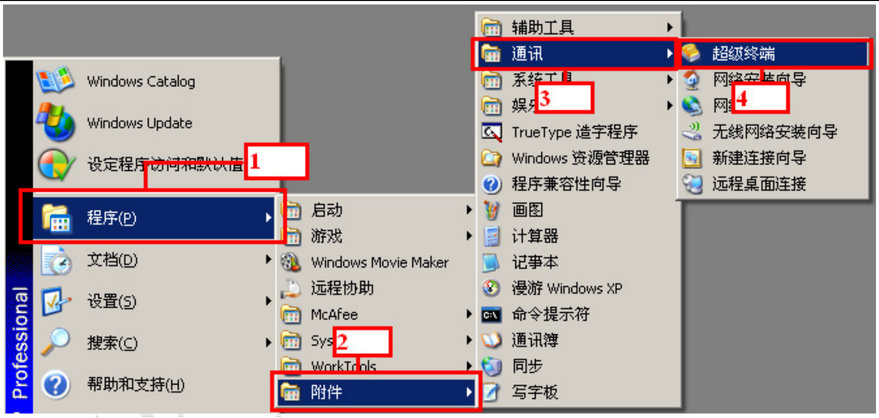
The following captured images might have some difference with the OS you have used in PC. If you have questions, please contact us.

2. 4. 1 Hyper-terminal configuration

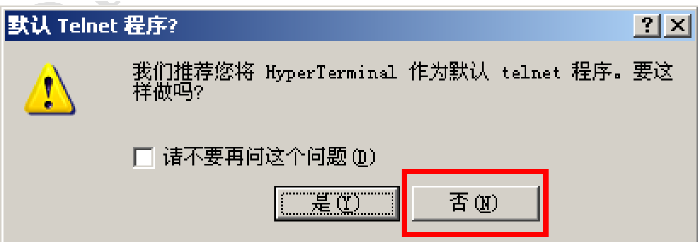
We recommend using Window self-carried hyper-terminal for interaction between PC and TQ2440. Here we introduce the configuration based on Windows XP hyper-terminal.

The following diagrams introduce the steps configuring the hyper-terminal:

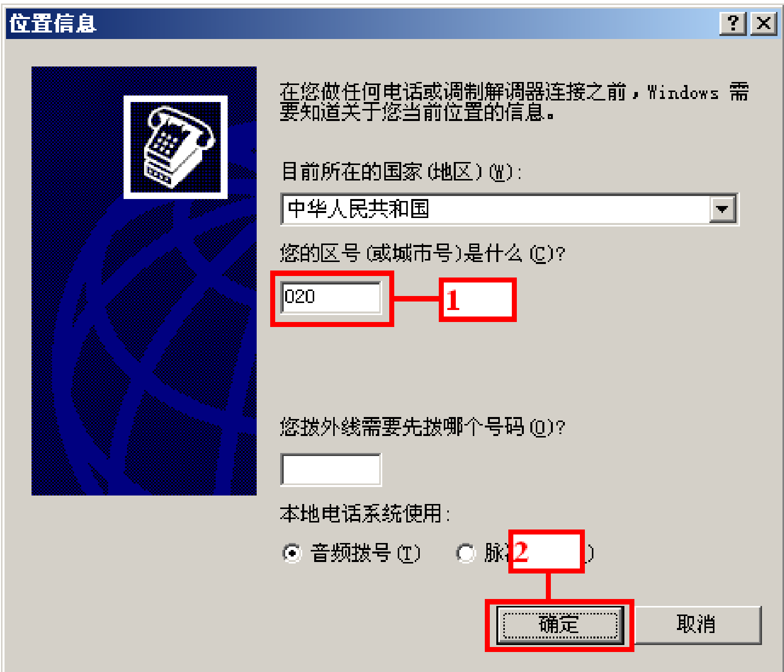
Step 1, open “Start->Programs->Accessories->Communications->Hyper Terminal”:



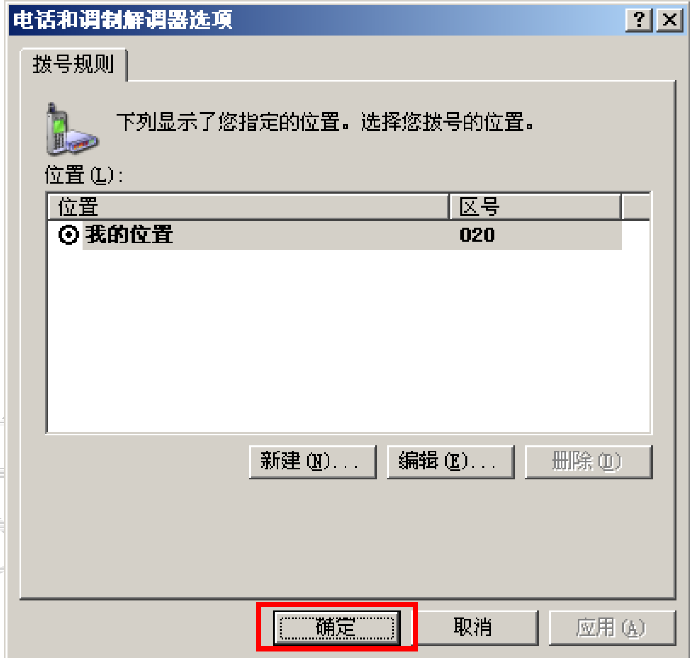
A pop-up window "Default Telnet Program" appears, choose "No":



A pop-up window "位置信息", fill in the blank "您的区号(或城市号)是什么？" with your district number, click "确定" and continue:

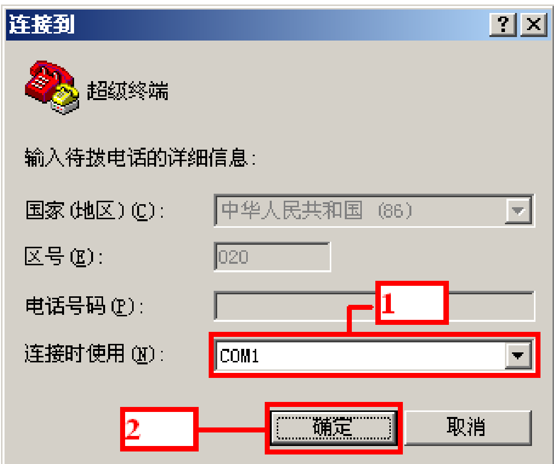


A dialog box as the following one appears, choose "确定":

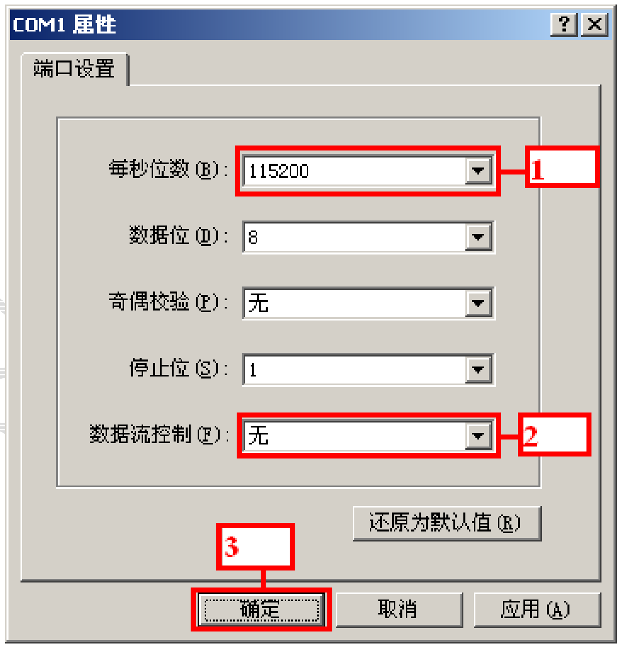


Step 2, an interface as the following one appears. Name your hyper-terminal and select an icon. Then click "确定" to continue:

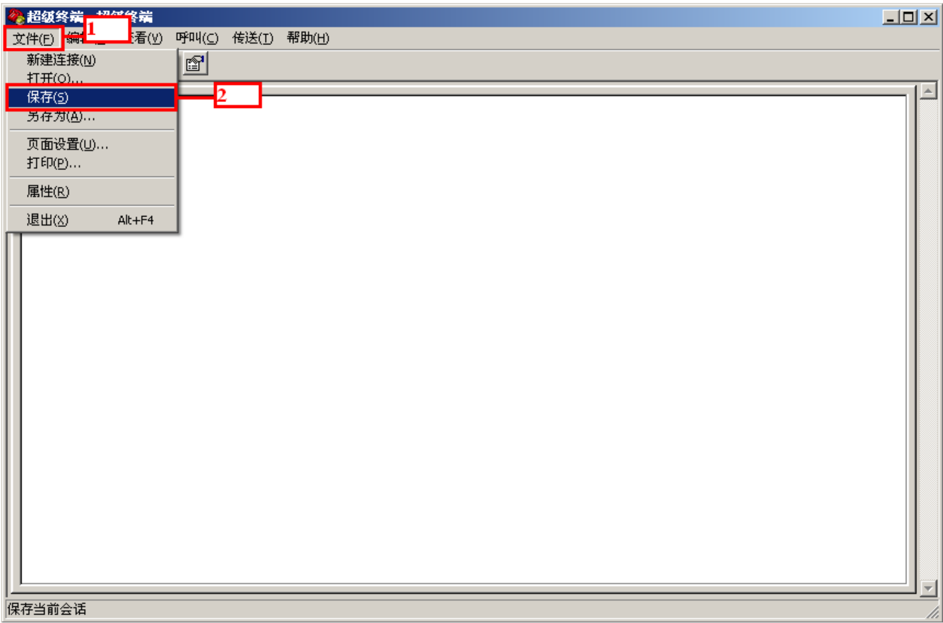
Step 3, an interface "连接到" as the following one appears. Select comN you connecting. N represents the serial port number of PC you are using. The example uses COM1 of PC. click "确定" and continue:



Step 4, an interface “COM1 属性” appears, setting: “波特率：115200，数据位：8，奇偶校验：无，停止位：1，数据流控制：无”. Click “确定” to continue:

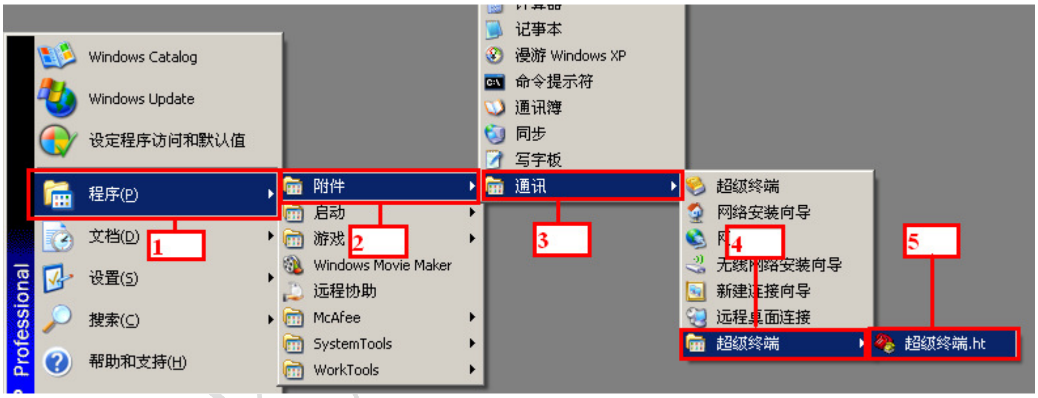


Step 5, the Typer Terminal window appears. Click menu “文件” and select “保存” to save your configuration:



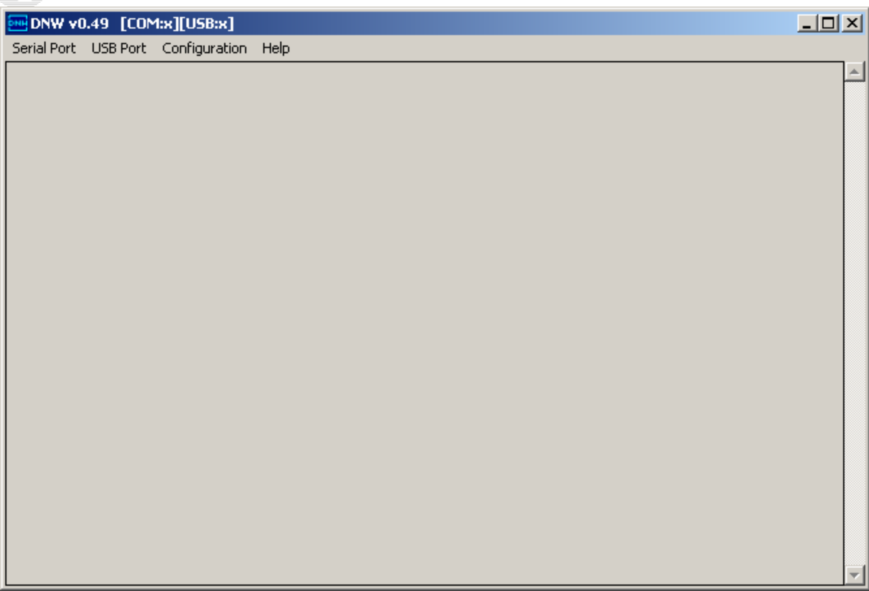
Step 6, in the future use, you can find the saved option “超级终端.ht” under “开始->程序->附件->通讯->超级终端”. Click it as follows:

(you can create a shortcut on your desktop for convenience.)

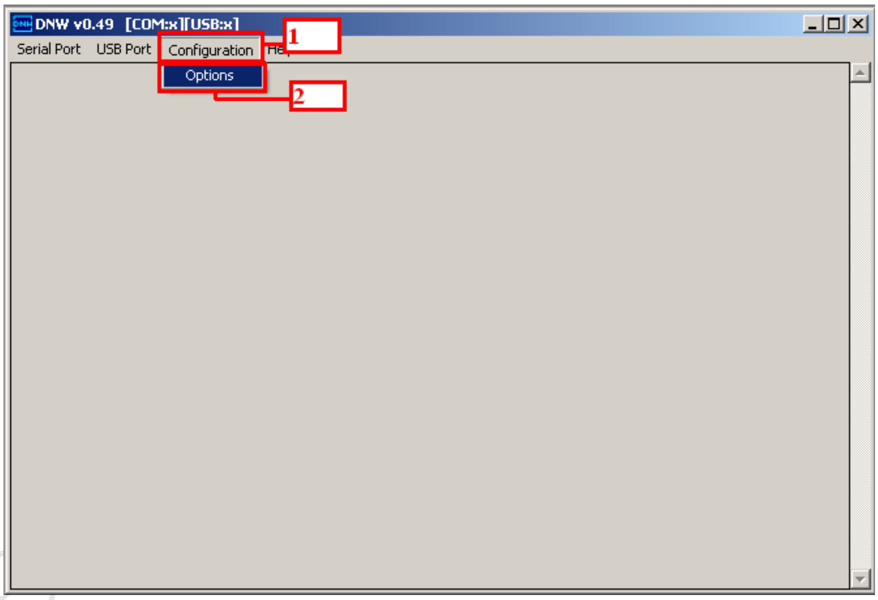


2. 4. 2 DNW software configuration

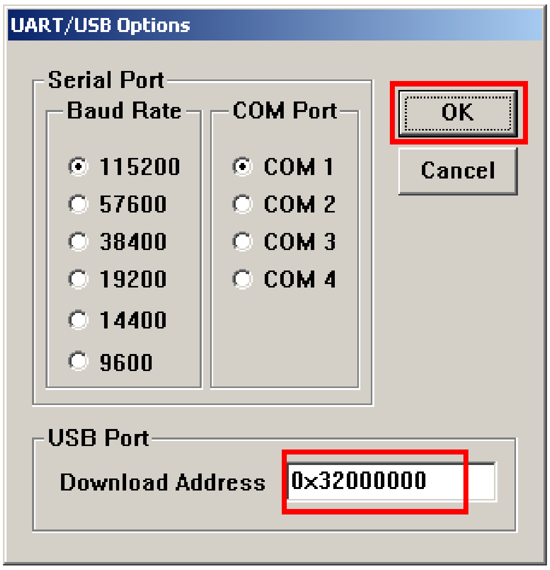
Find DNW software under directory “Windows 平台工具\DNW”. Double-click to open it:



Step 1, click “Configuration -> Options”, the configuration table “UART/USB Options” appears.



Step 2, choose “115200” of “Baud Rate”, choose “COM1” of “COM Port” (choose the right one according to actual situation), fill in “0x32000000” of “USB Port”, click “OK” to finish the DNW configuration:

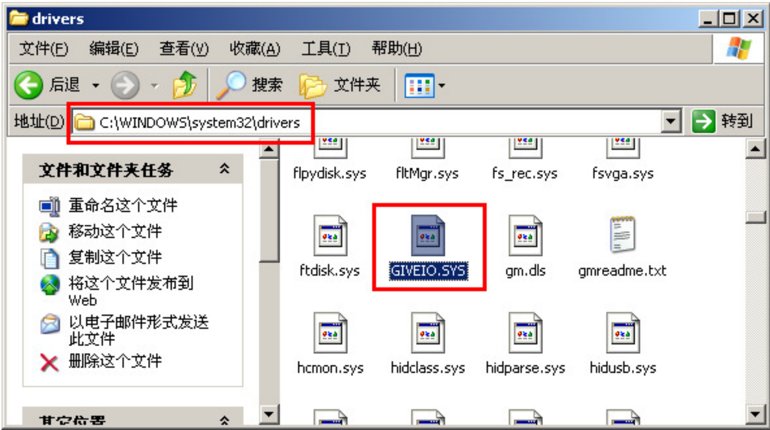


2. 4. 3 GIVEIO driver intallation

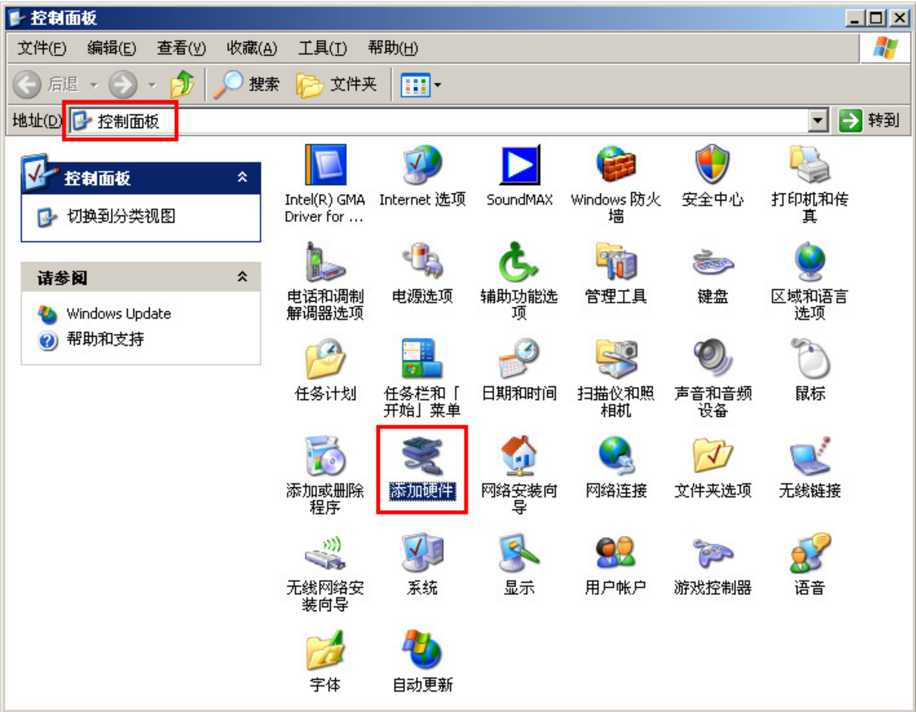
If the user wants to burn u-boot with the Jtage software SJF2440.exe, a driver needs to be installed to virtualize parallel port into IO port. Pay attention to the parallel port configuration under BIOS, SPP and EPP mode is recommended, but ECP mode is not recommended.

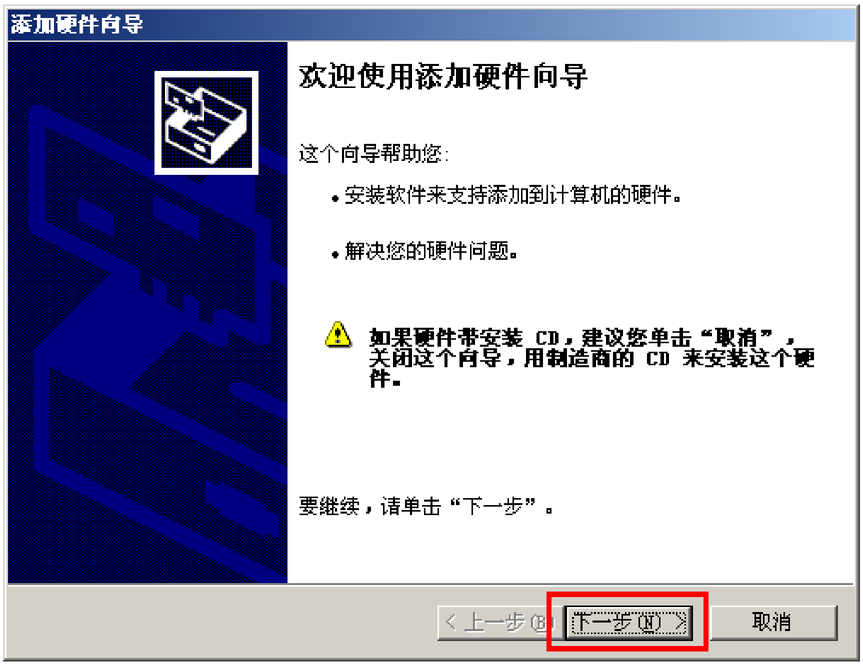
The steps how to install GIVEIO is shown in the following.

Step 1, Find giveio driver in the CD under the directory “Windows 平台工具\GIVEIO”. Copy the file “giveio.sys” to your system disk, under the directory “WINDOWS\system32\drivers”

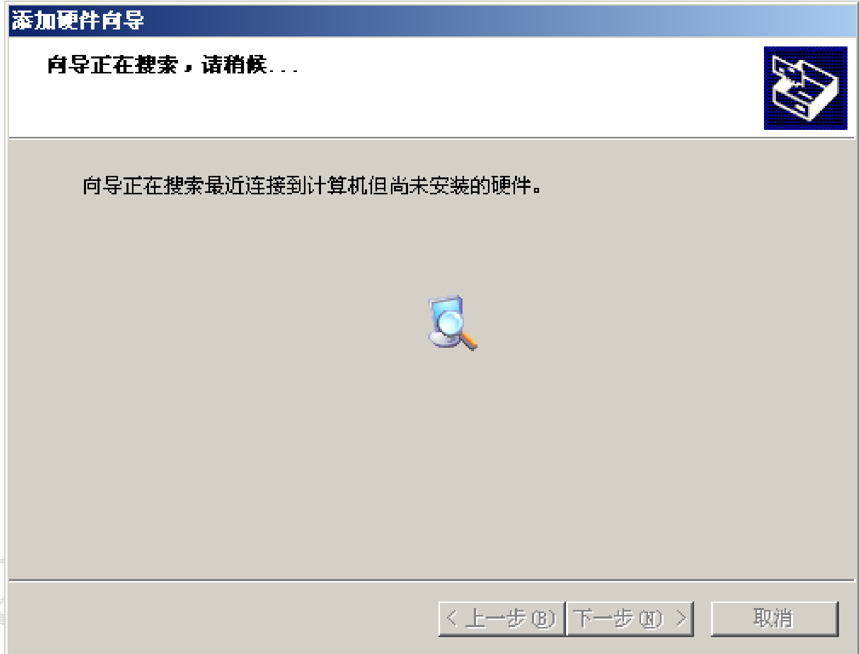


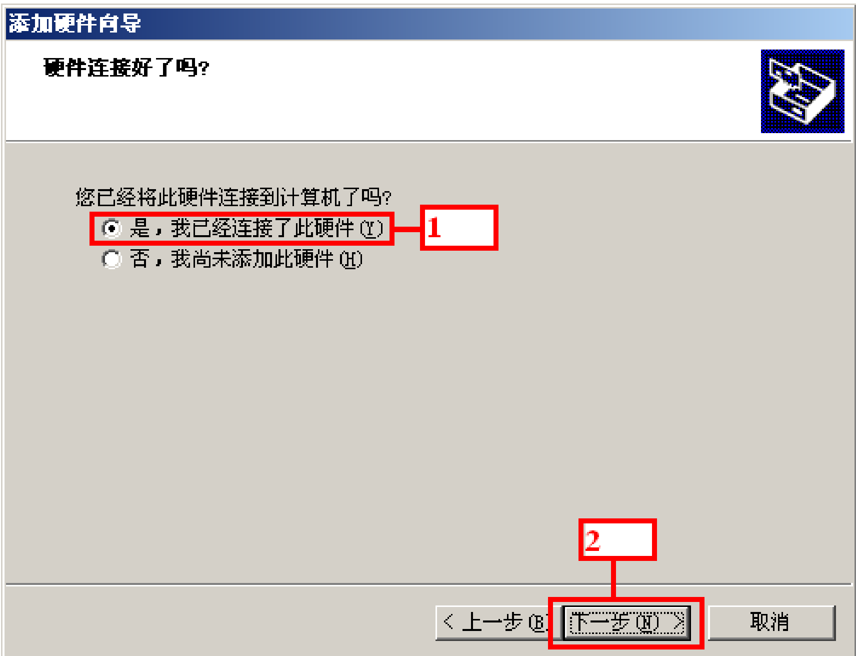
Step 2, open “控制面板” on your PC, double-click the icon “添加硬件” and enter the interface 添加硬件, click “下一步” to continue:



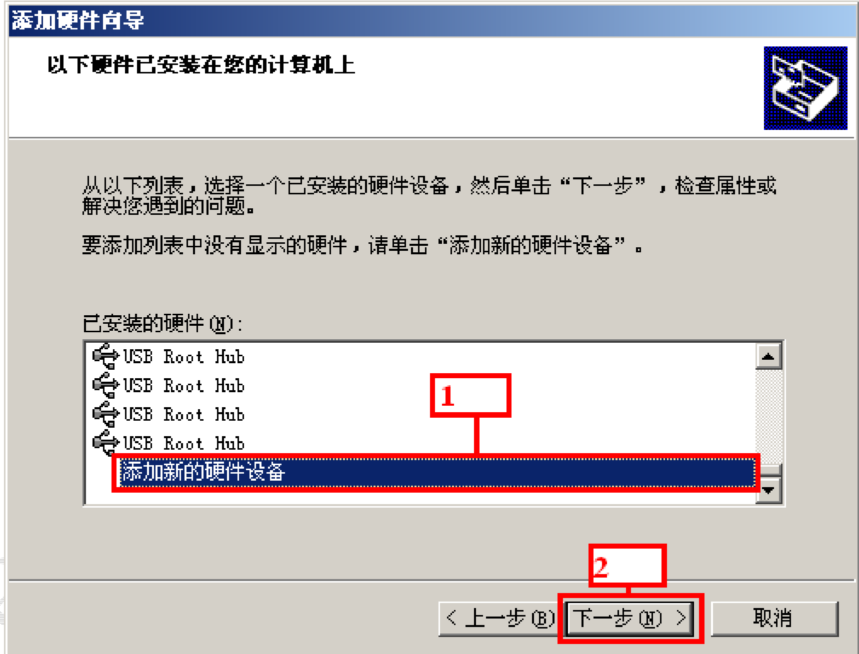


Step 3, system will find the hardware automatically. When the search is finished, the next window appears. Select “是，我已经连接了此硬件”, and click “下一步” to continue:

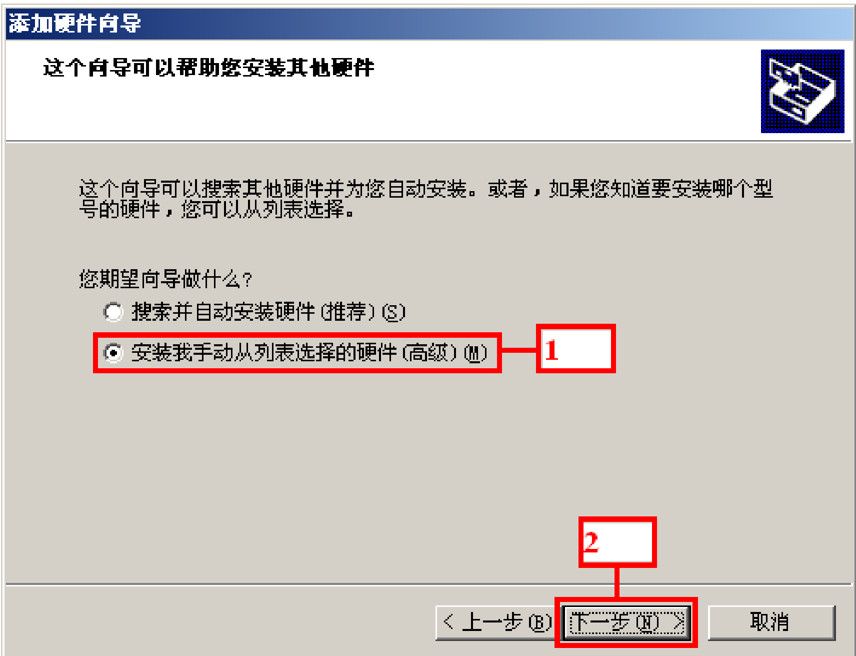




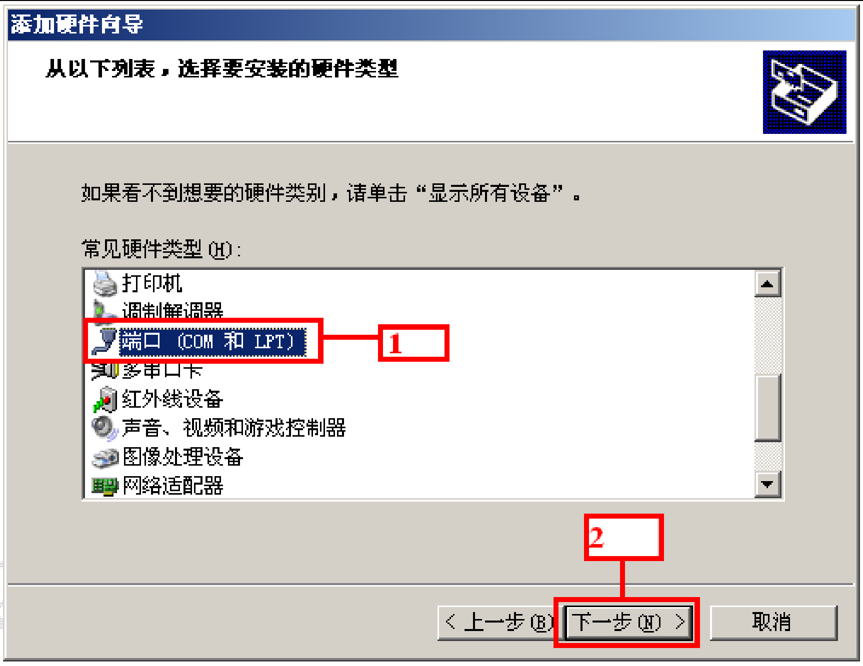
Step 4, select the option as the following one in the appearing interface and click “下一步” to continue:



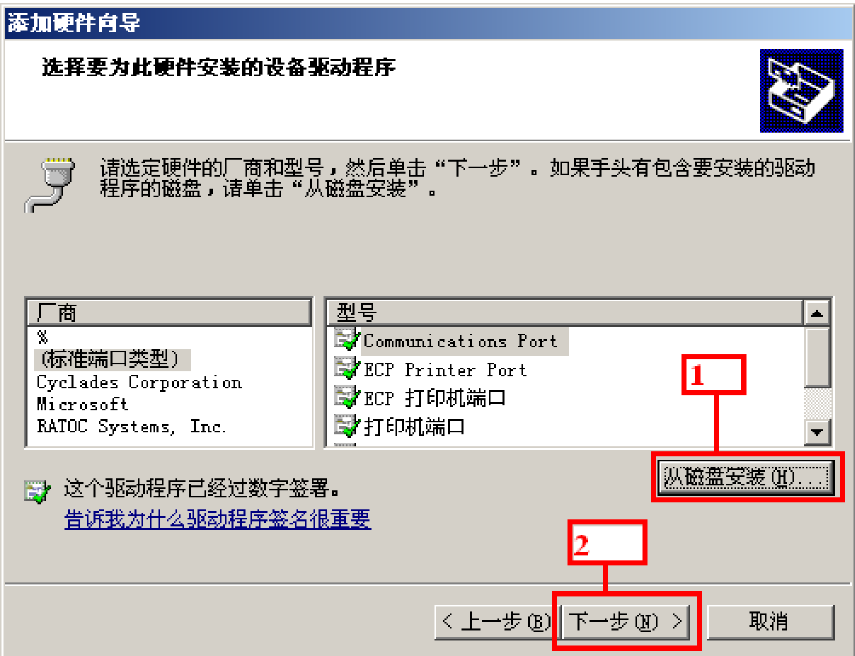
Step 5, the interface “安装向导” appears. Select “安装我手动从列表选择的硬件(高级)” and click “下一步” to continue:



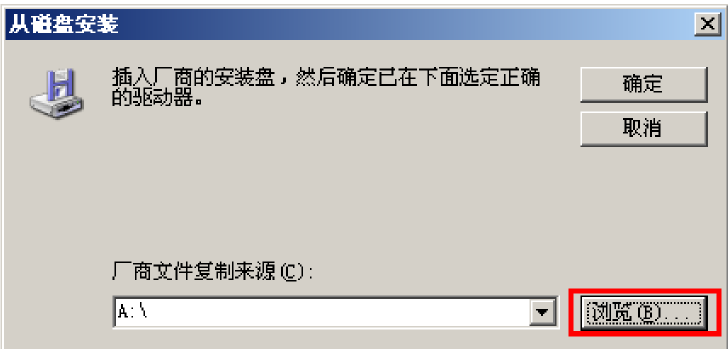
Step 6, select “端口(COM 和 LPT)” among the hardware list and click “下一步” to continue:



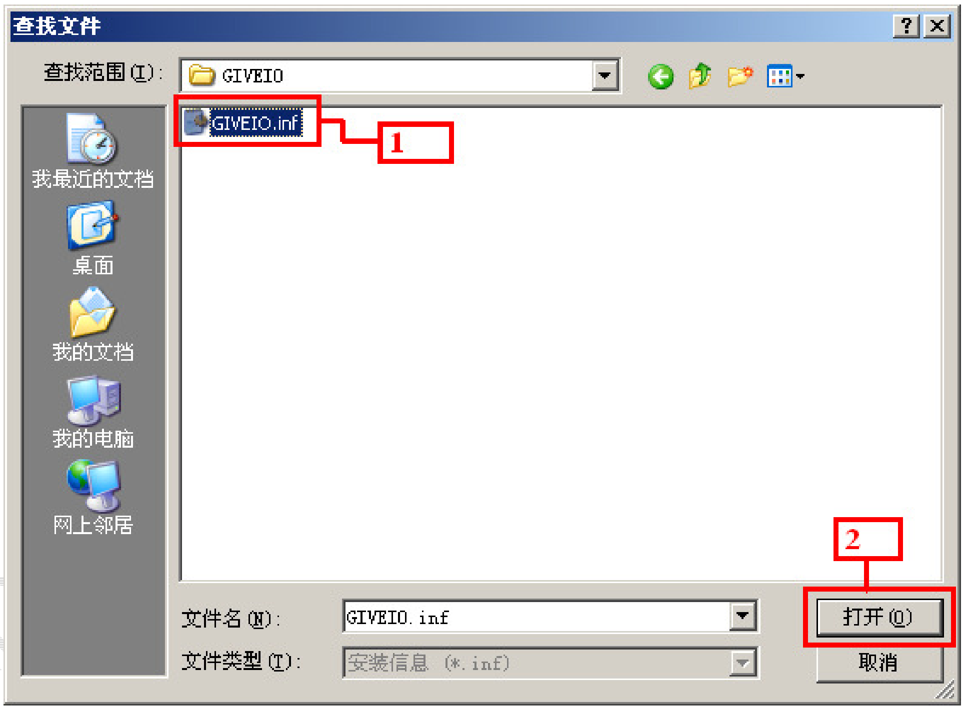
Step 7, click “从磁盘安装” and continue:



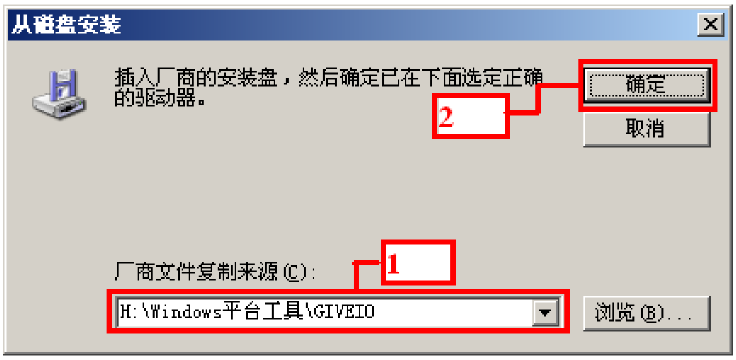
Step 8, select “浏览” in “从磁盘安装” interface:



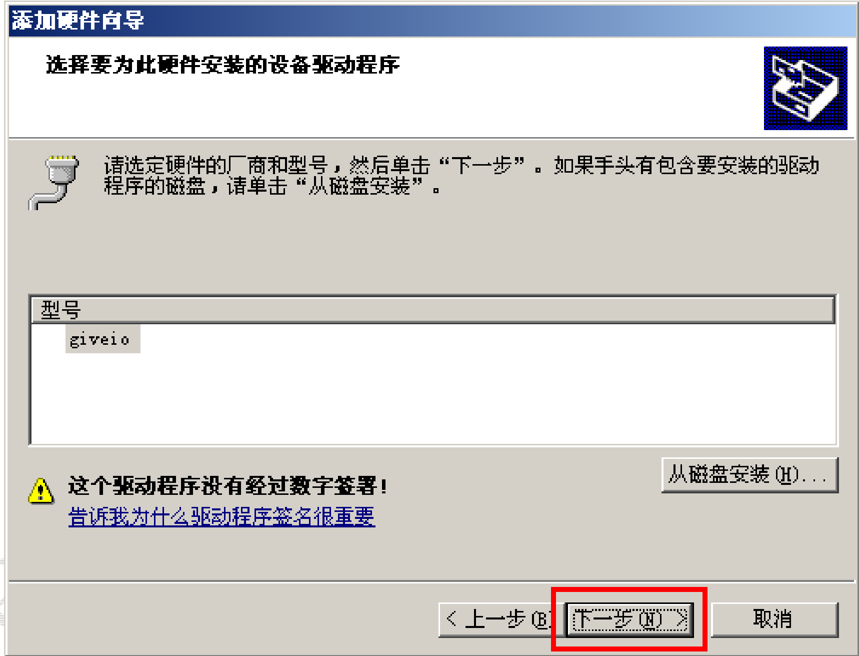
Locate the previous GIVEIO directory. Find the file “GIVEIO.inf” and click “打开” to continue:



Back to the interface “从磁盘安装”, and click “确定” to continue:

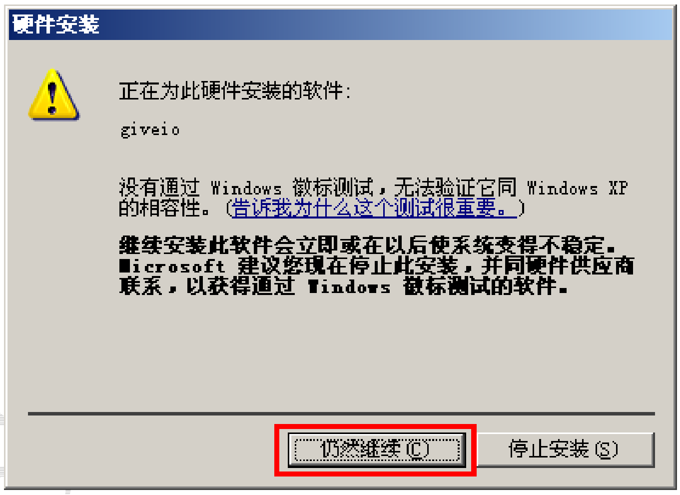


Step 9, back to the interface concerning installing device drivers. Select “giveio” device and click “下一步” to continue:

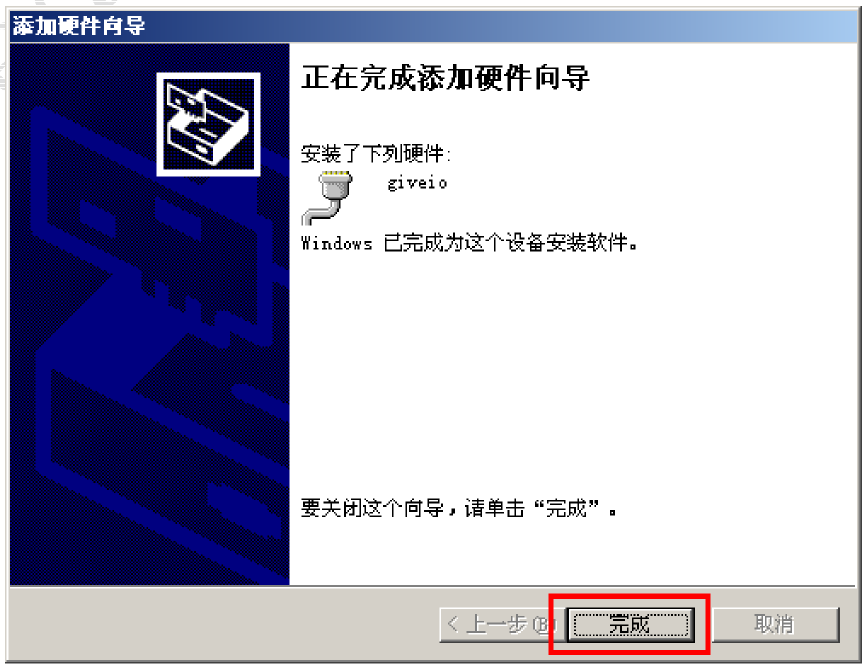


Step 10, “向导准备安装您的硬件” interface appears. Click “下一步” to continue:

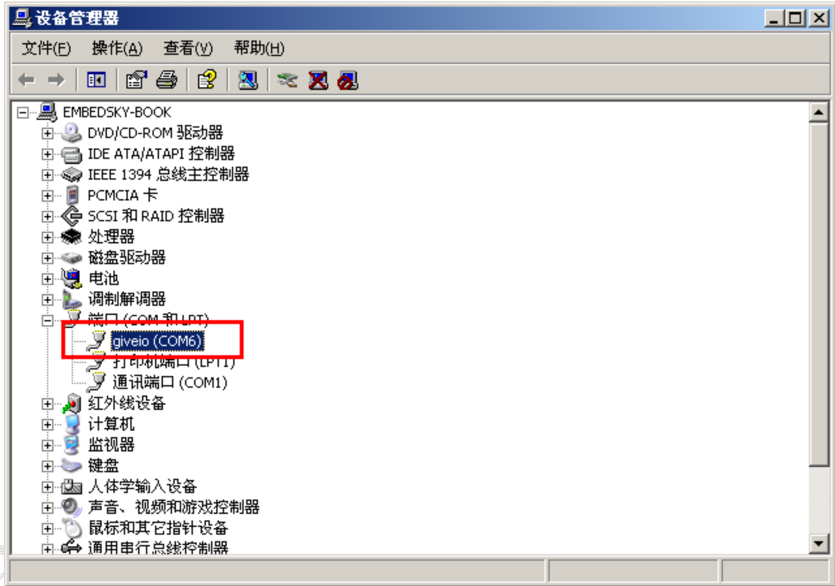
An interface appears warning that the driver has not been authenticated by Microsoft. Click “仍然继续” to continue:



Step 11, click “完成” to finish the installation:

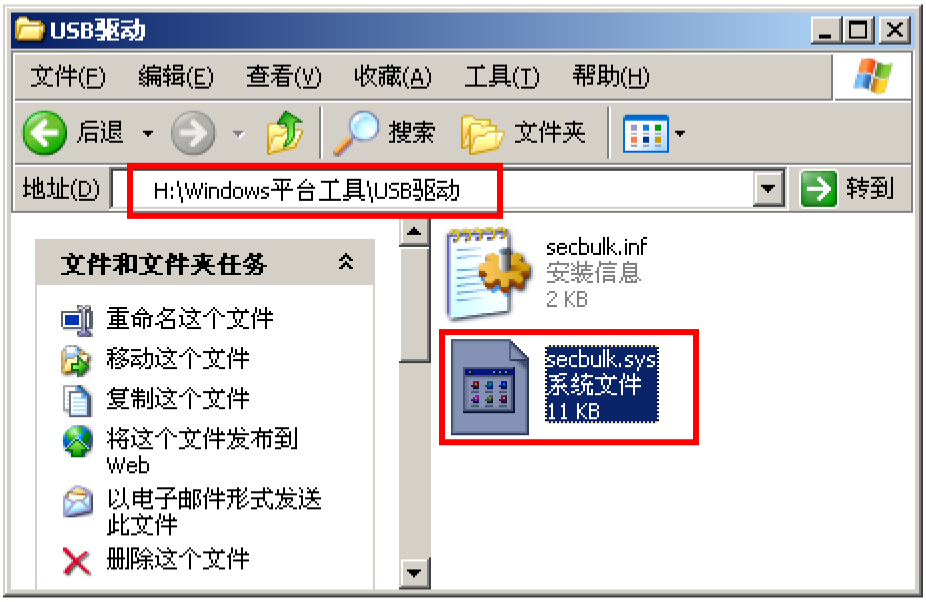


Step 12, the newly installed device could be found in “设备管理器”:

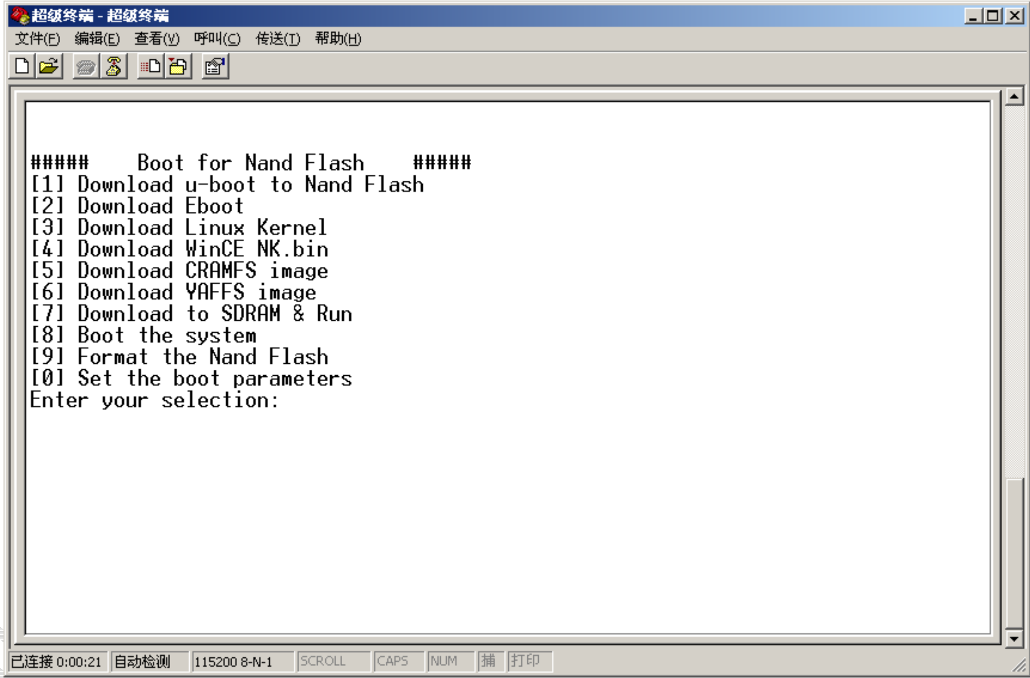


2. 4. 4 USB download-driver installation

The following steps introduce how to install USB download-driver. The driver is located under the directory "Windows 平台工具\USB 驱动":

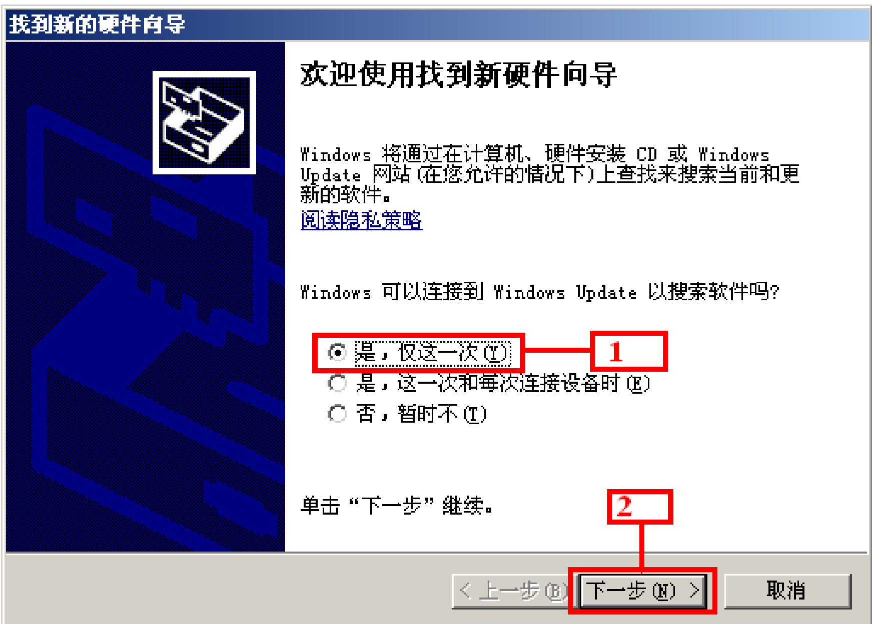


Step 1, open hyper-terminal, and link the serial port line and power line; press the space-key of PC and hold, and Switch on the power. The hyper-terminal will display the u-boot console (instruction: USB download-driver needs to be installed in u-boot console).

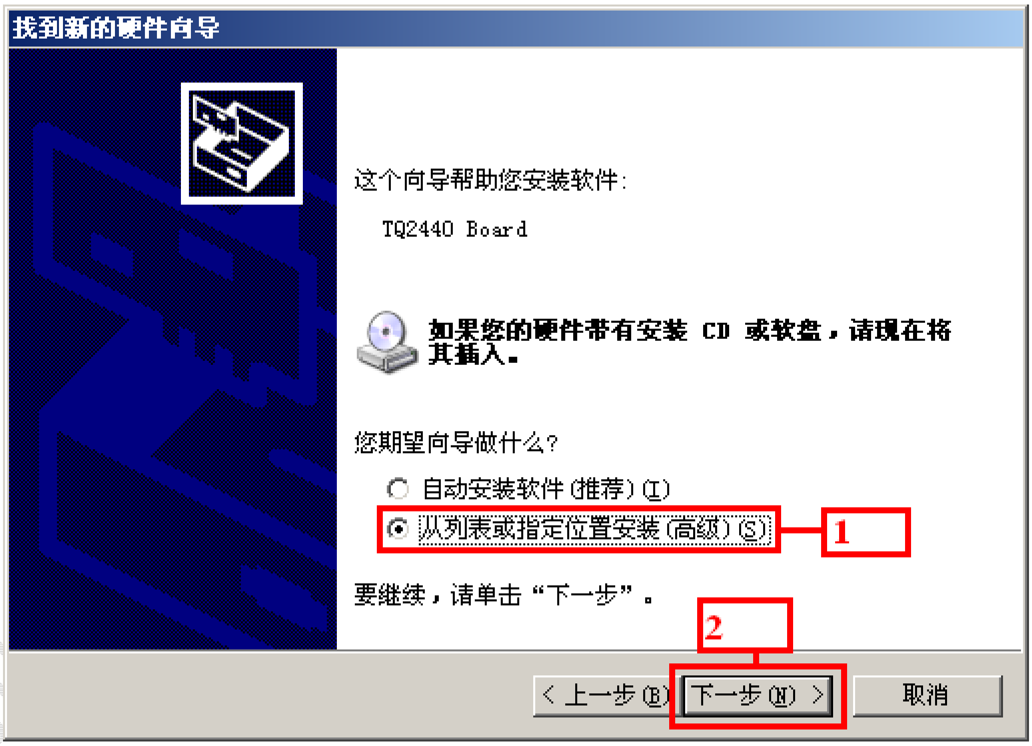


Step 2, when linking the USB wire, Windows XP can recognize the new device automatically as the following diagram:

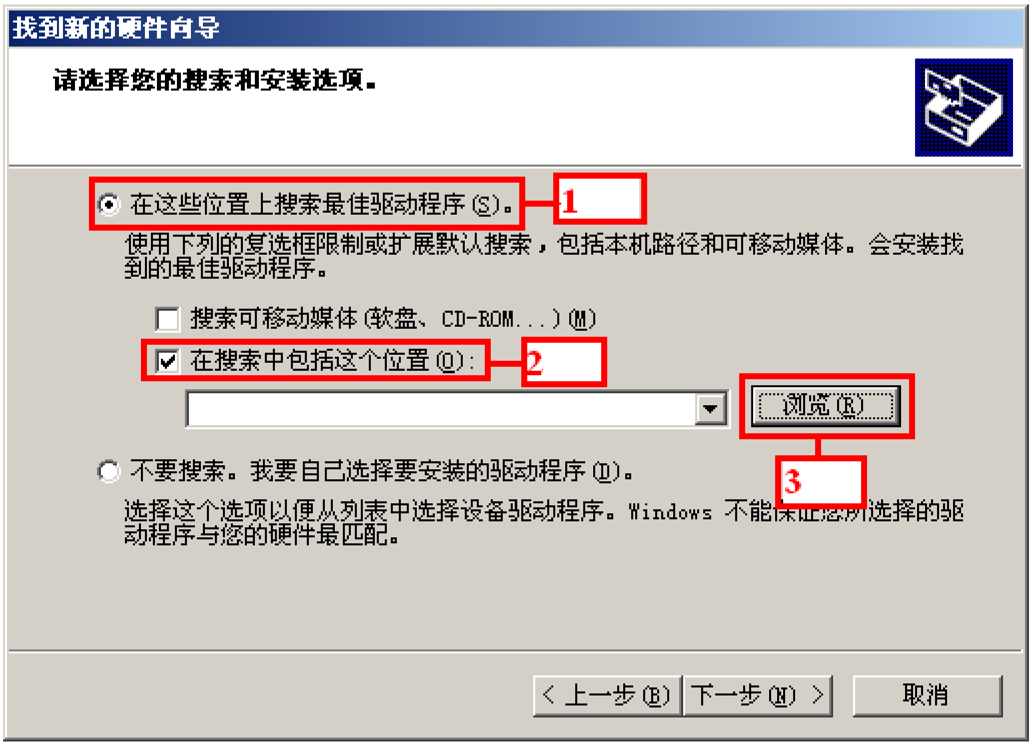
The interface “找到新的硬件向导” pops up. Select “是，仅这一次” and click “下一步” to continue:



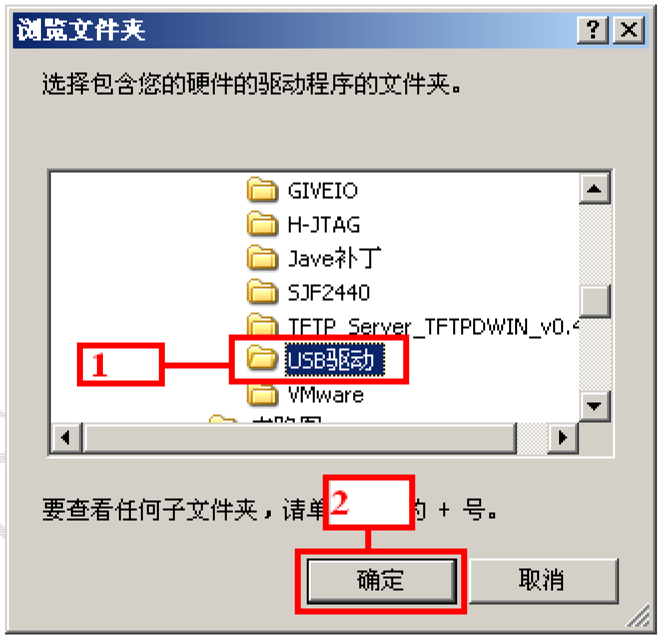
Step 3, select “从列表或指定位置安装(高级)” in the next interface and click “下一步” to continue:

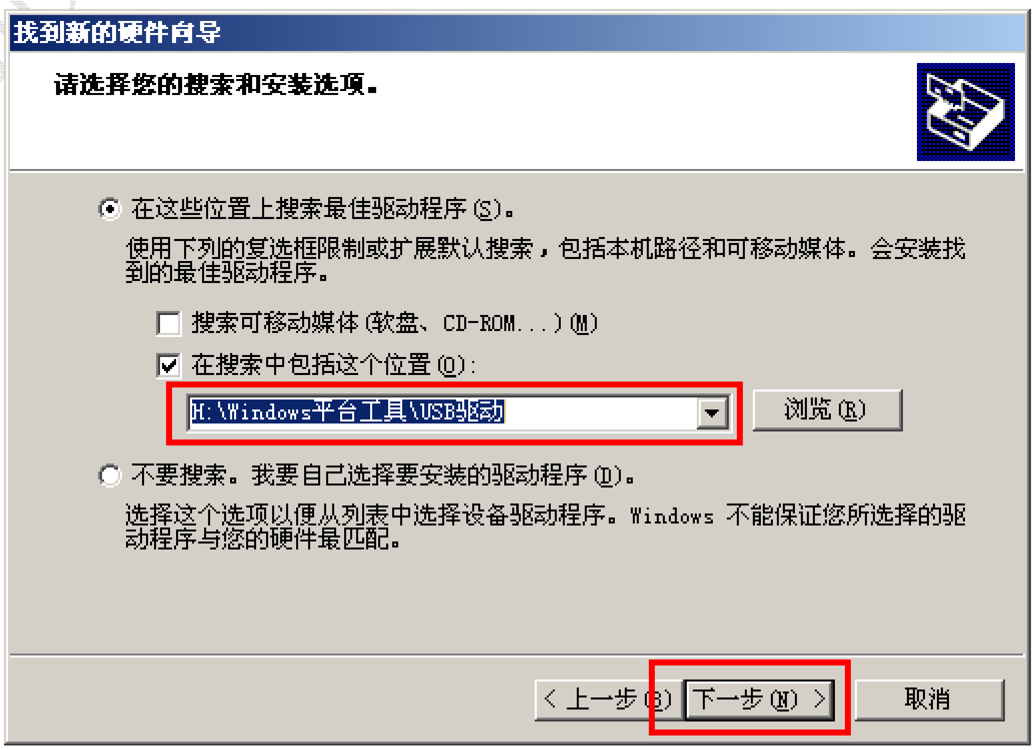


Step 4, select "在搜索中包括这个位置" in "在这些位置上搜索最佳驱动程序" menu and click "浏览":



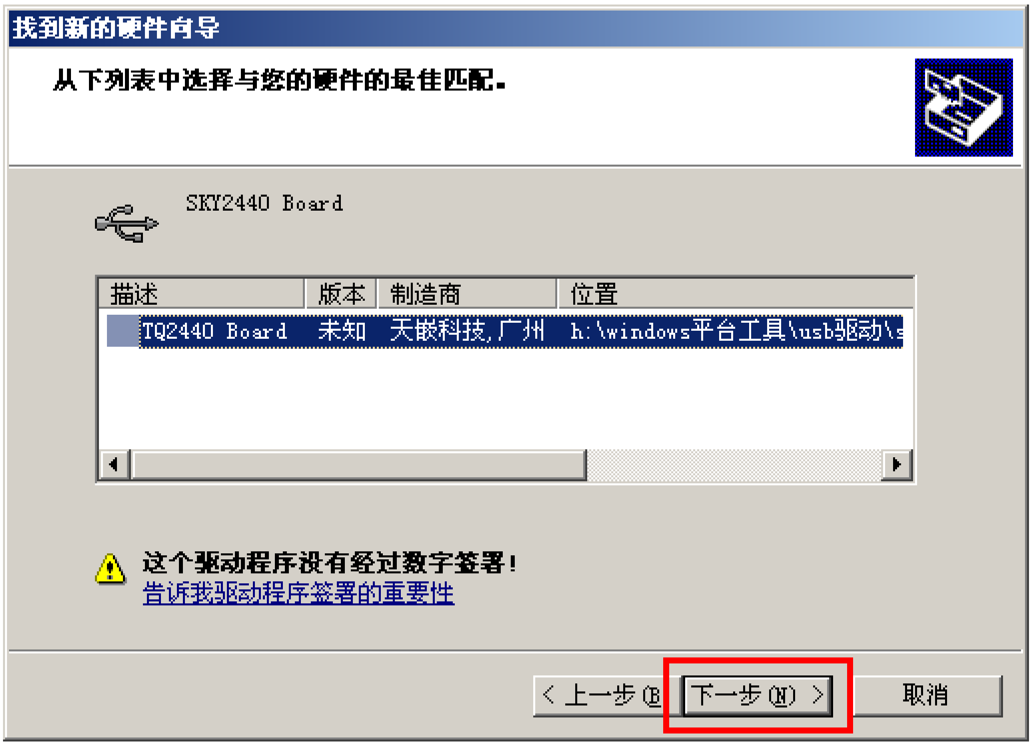
Locate the driver and click "确定" to go back to the upper diagram. Click "下一步" to continue:



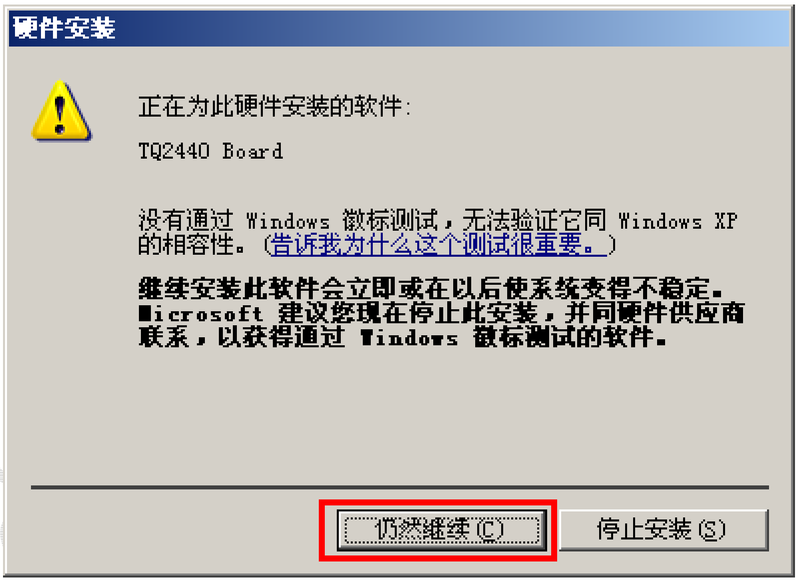


Step 5, the installing guide begins to search hardware device:

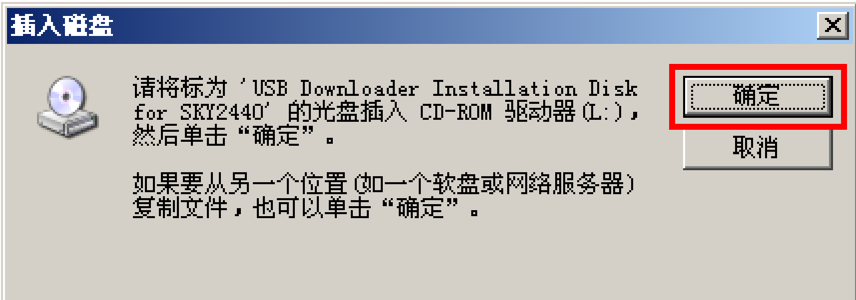
The following diagram appears. Select “SEC SOC Test Board” and click “下一步” to continue:



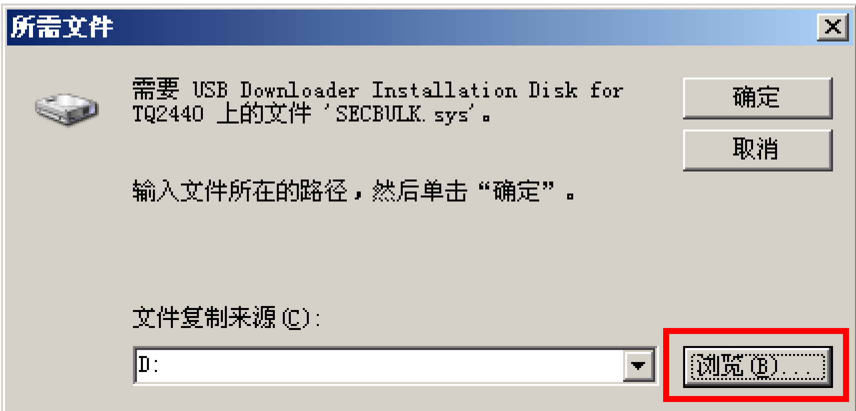
Step 6, when installing the driver, the following interface appears. Click “仍然继续” to continue:



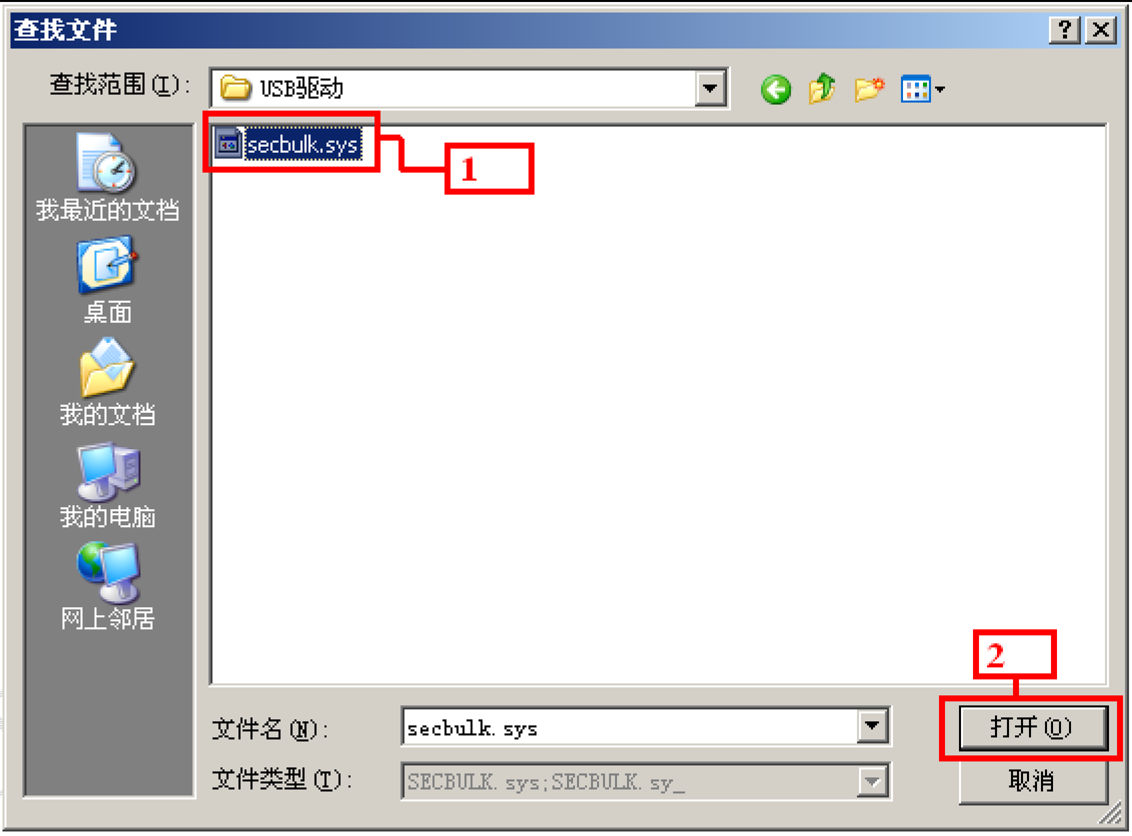
Step 7, the interface “所需文件” appears:



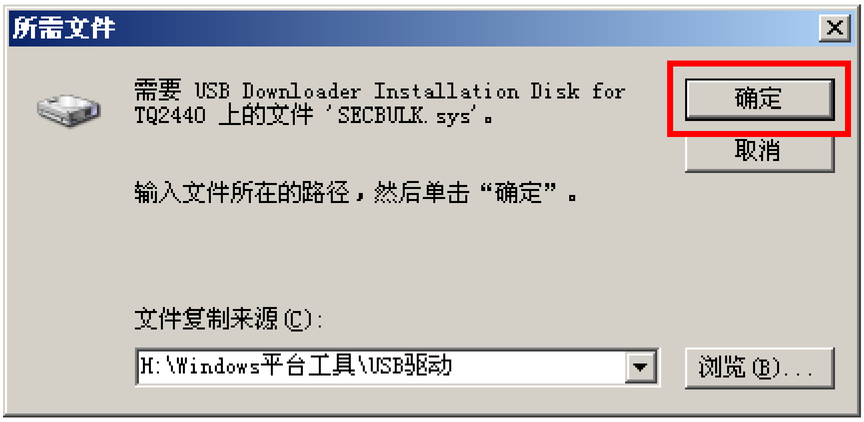
Click “确定”. Then the following interface appears. Click “浏览” to locate the driver:



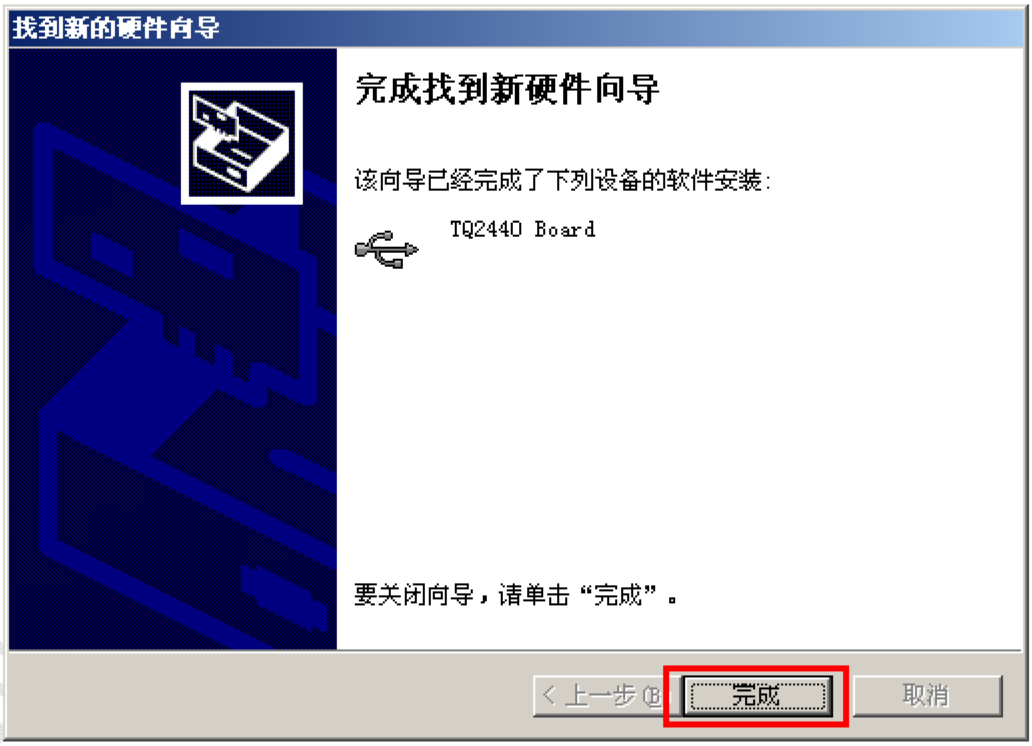
Locate the file “secbulk.sys”, and click “打开” to continue:



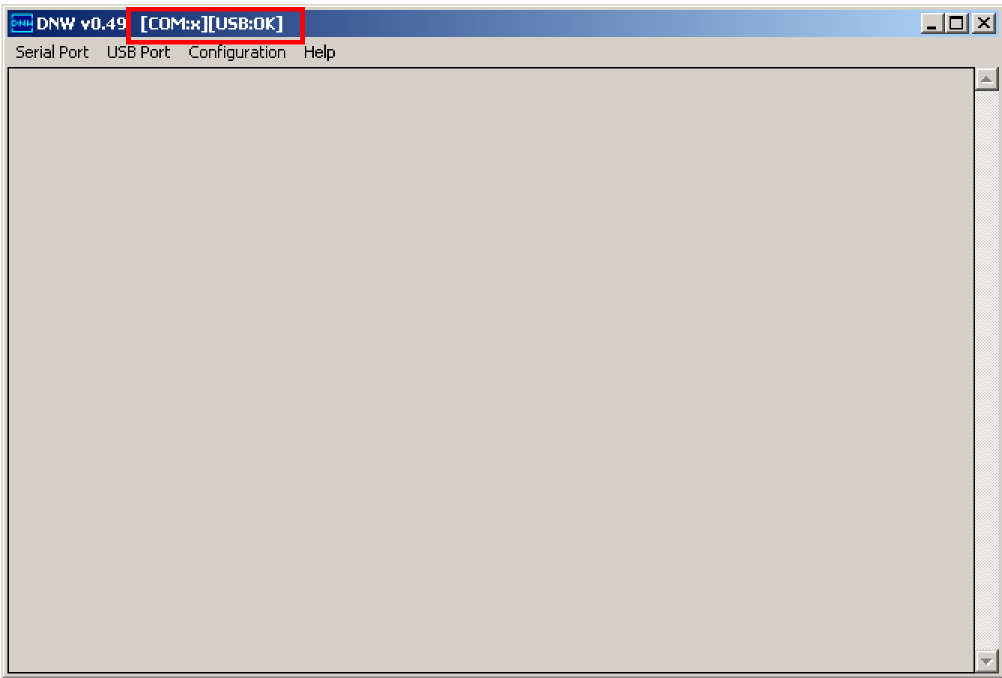
Back to "所需文件" interface, click "确定" to continue:



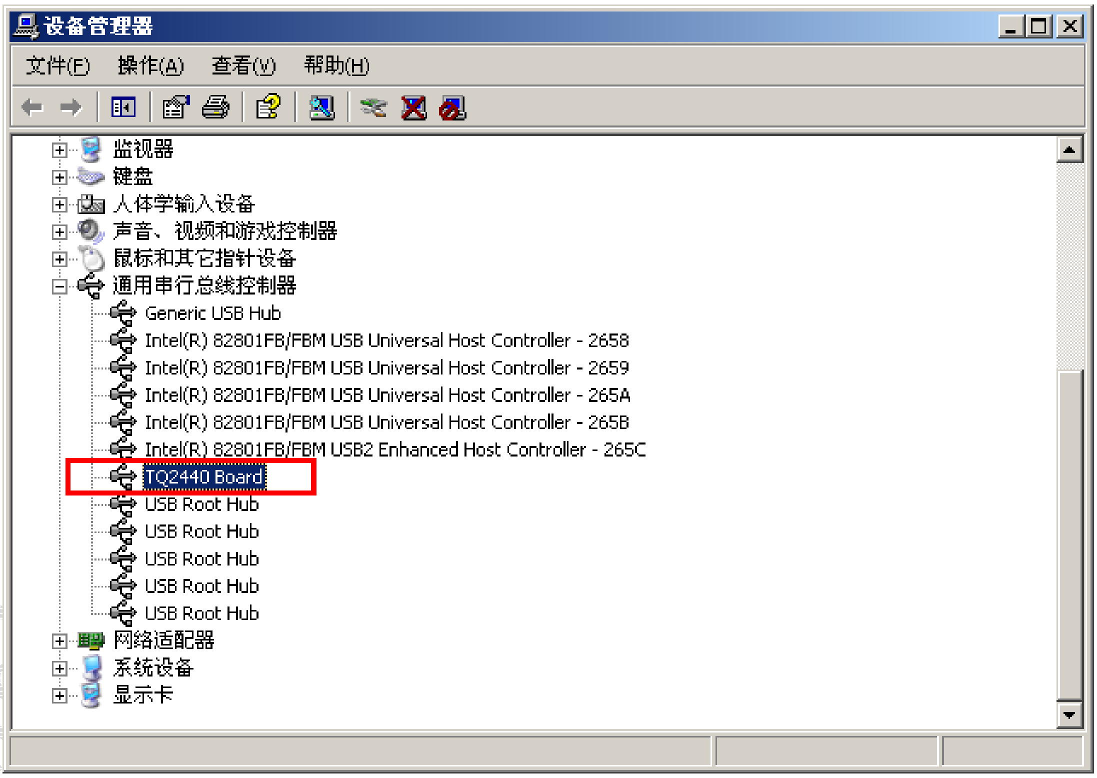
Step 8, click "完成" to finish USB download-driver installation:



Step 9, after the USB download-driver has been installed, open DNW software. The mark “[COM:x][USB:OK]” could be found on top of the window:



The USB driver installed previously could be found in “设备管理器”:



Now the user can use USB to download u-boot, operating system and file system.

2. 5 Linux environment configuration

It is suggested to refer to《TQ2440 之 RedHat9 安装 step by step》before read this part

Instruction: In the latter Linux operation, the PC command line is executed on ordinary terminal; while the platform command line is executed on hyper-terminal. We attach symbol "#" to each PC operation instruction in order to distinguish these 2 kinds of command line. Please be aware of the difference.

The Linux compression package must be decompressed under Linux. The filename and command are case sensitive which is different from Windows. Please be aware of that.

2. 5. 1 Build cross-compile environment

The cross_compiler is the main environment under Linux. We introduce a process building a development environment which can compile arm-linux kernel, driver and application under RedHat 9.0.

Copy the compression package "crosstools_all.tar.bz2" from the directory "Linux" to the directory "/opt/EmbedSky/" in Linux system, and decompress the package under the current directory: (the following commands are executed in PC).

#cd /opt/EmbedSky

#tar xvfj crosstools_all.tar.bz2 -C /

After the upper operation, the compiler has been installed under the directory "2.95.3" and "3.3.2" of "/usr/local/arm/" and the directory of "/opt/EmbedSky/crosstools_3.4.1_softfloat/". The makefile will be installed automatically under the directory "/usr/local/sbin/":

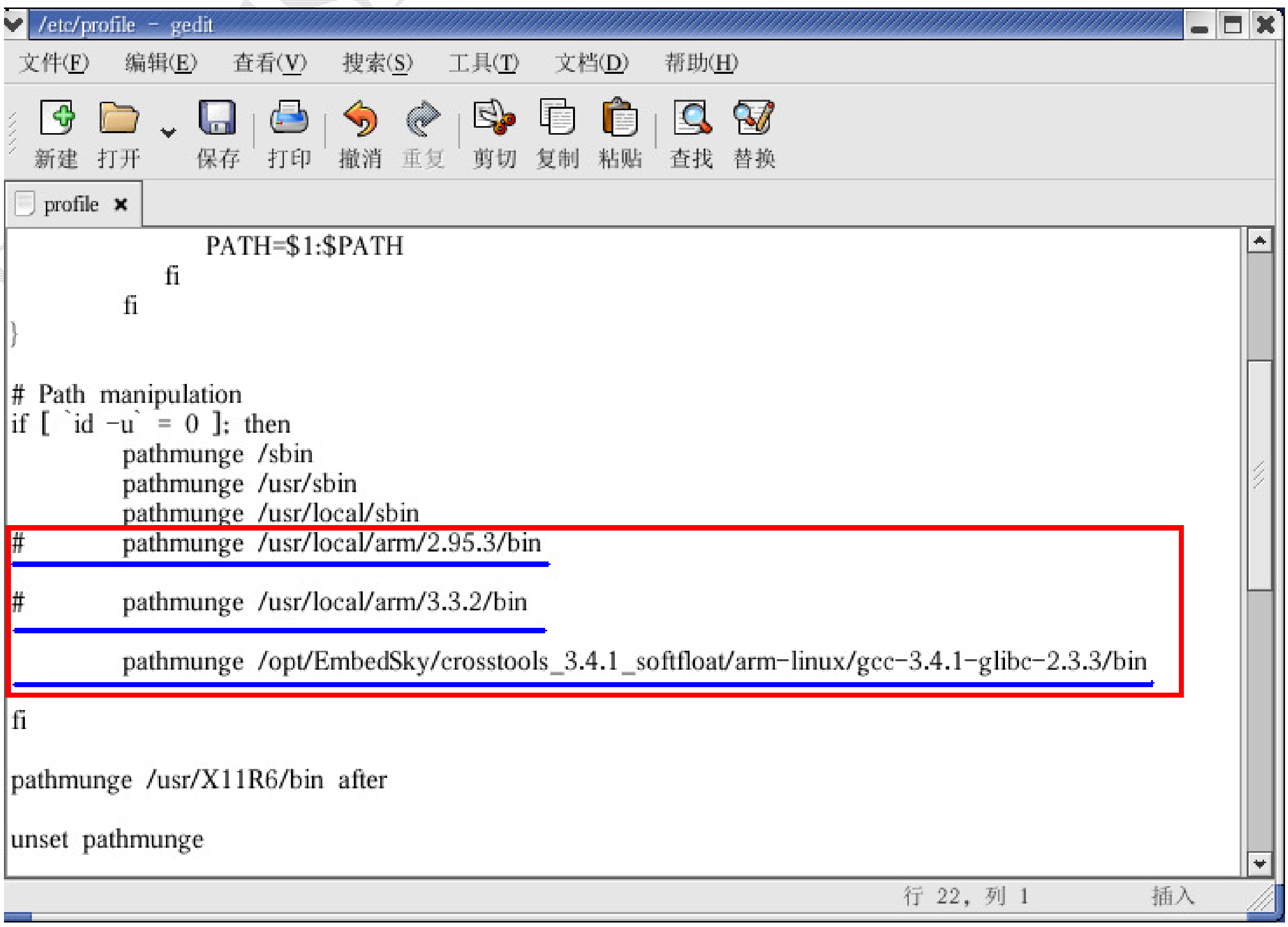
- The cross-compile compiler of 3.3.2 version is used to compile Qtopia/Embedded.
- The cross-compile compiler of 2.95.3 is used to compile VIVI and transplant boa.
- The cross-compile compiler of 3.4.1_softfolat is used to compile kernel, busybox, u-boot and application.

Execute the command:

#gedit /etc/profile

Add the following information in "profile":

(the following frame contains the added information. If the user tries to use the cross-compiler of a certain version, please remove its prefix "#", and add "#" to the head of other versions. The lines highlighted with blue underline as following diagram is required to be added the prefix "#")



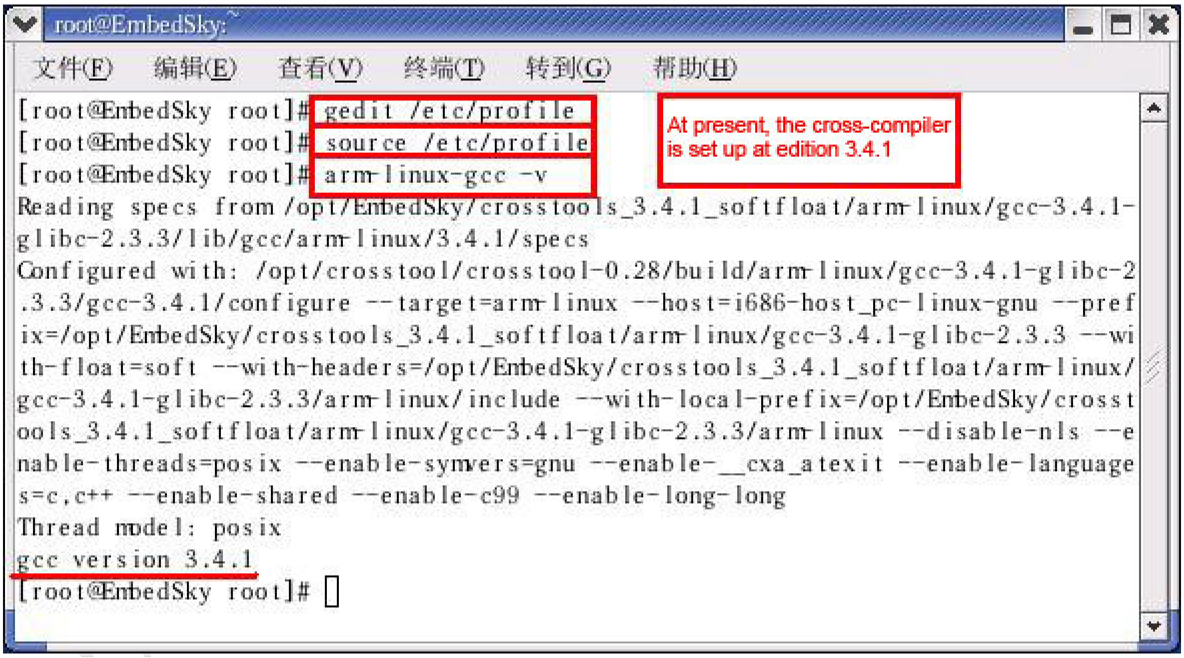
Save "profile" and execute the following command to set a default cross-compiler:

#source /etc/profile

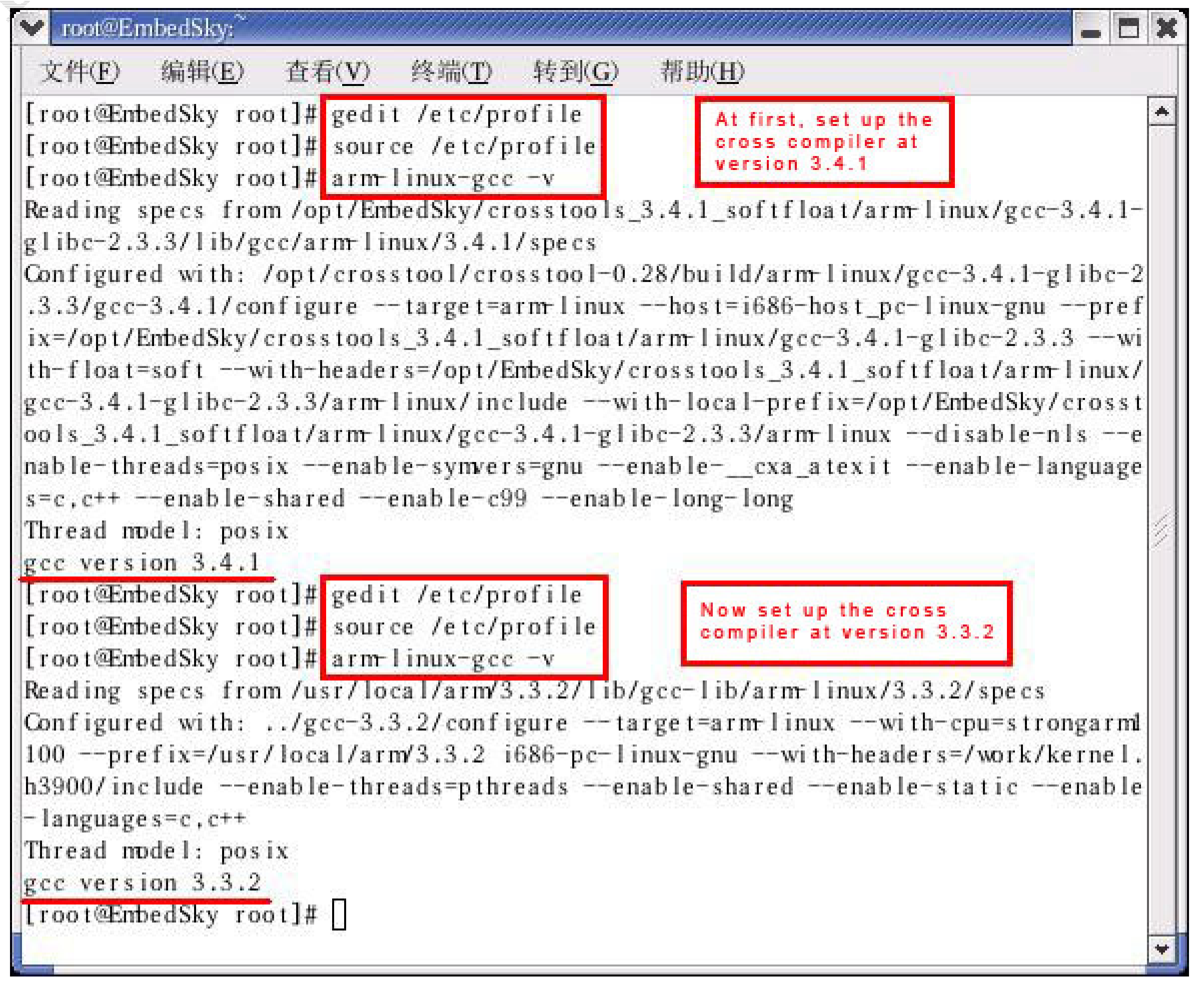
Execute the following command to check if the cross-compiler has been installed successfully and check the revised version:

#arm-linux-gcc -v

Get the following information:



The cross-compiler version might be frequently changed. Use the previous command "gedit /etc/profile" to modify the file "/etc/profile", and use it to validate "source /etc/profile". Then execute the command "arm-linux-gcc -v" to check the revised version of cross-compiler.



(caution: After update, you might find the version is not changed in current terminal. You can open a new terminal to check the revised version.)

2. 5. 2 Network File Service (NFS) configuration

When installing RedHat 9.0, if you choose complete installation, all the relevant components will be installed by default. Configure the Network File Service as the following diagram:

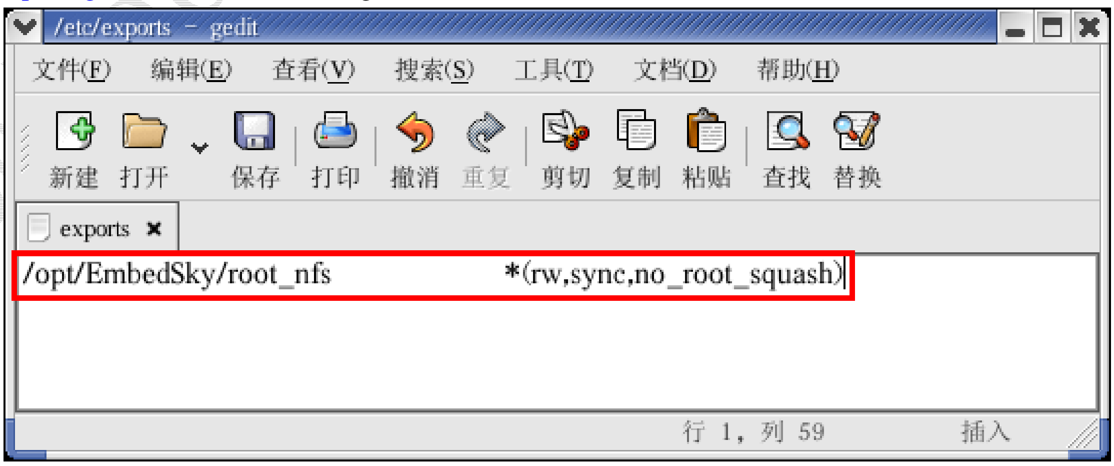
(caution: Close the firewall of Linux as following diagram, otherwise the NFS can't work possibly)

Use the following command to close firewall:

#/etc/init.d/iptables stop

Configure the sharing directory:

Execute the command "#gedit /etc/exports", edit the configuration files of NFS (caution: The file is empty in the first opening), and add the following contents:



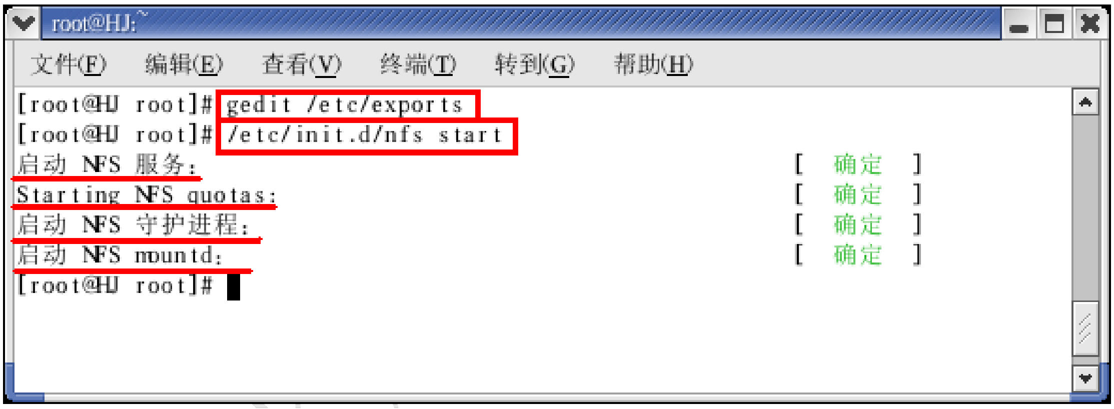
- "/opt/EmbedSky/root_nfs" represents NFS sharing directory, it can be used as root file system of platform and be mounted by NFS:
- "*" represents that all the customers have the right to mount this directory.
- "rw" represents that the customer which mounting this directory is authorized to write and read this directory.
- "no_root_squash" represents that the customer who mount this directory could be treated as the root of server.

Build the sharing directory:

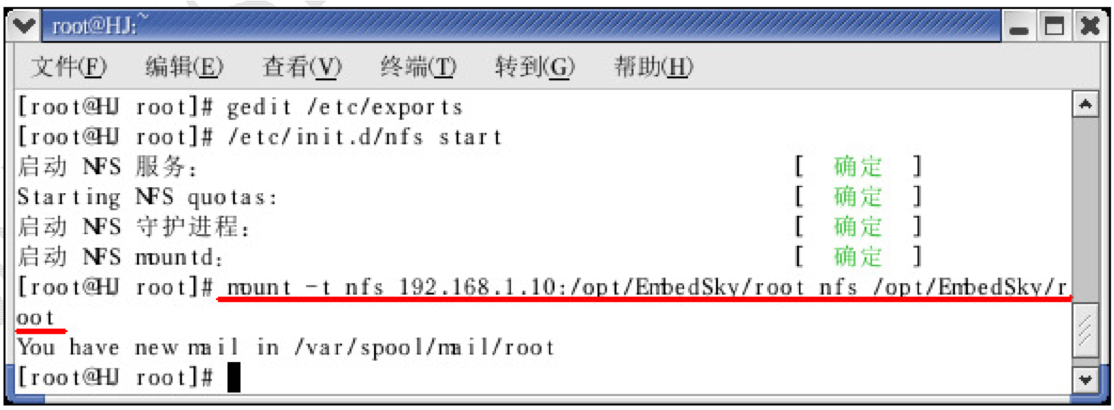
Find the file "root_nfs.tar.bz2" under the directory "Linux" in CD, and decompress it under the directory "/opt/EmbedSky/" in linux. The decompression command is "#tar xvfj root_nfs.tar.bz2 -C /".

Start and stop NFS:

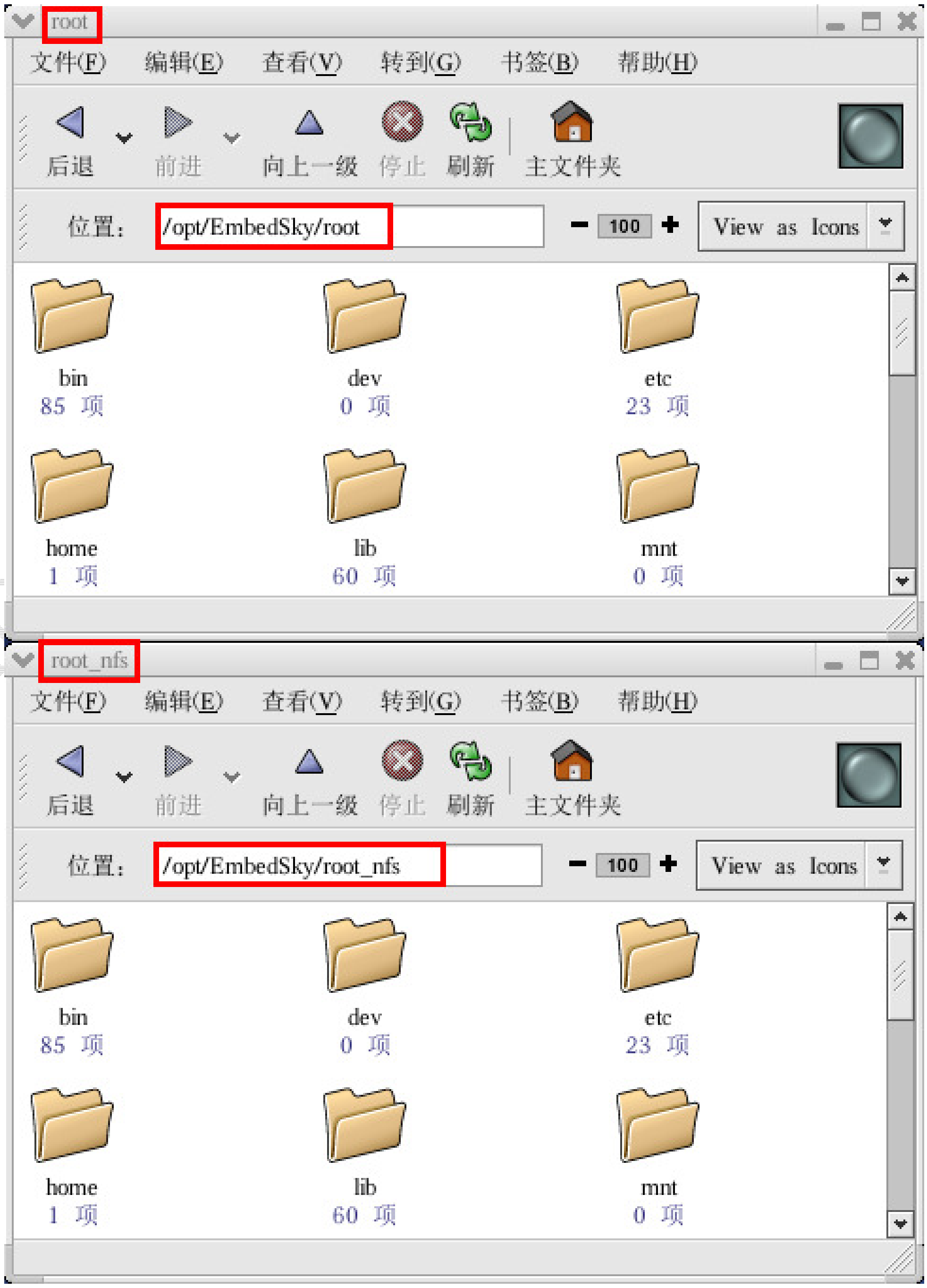
Execute the command "#/etc/init.d/nfs start" to start NFS:



Use the following command to check the NFS running or not:



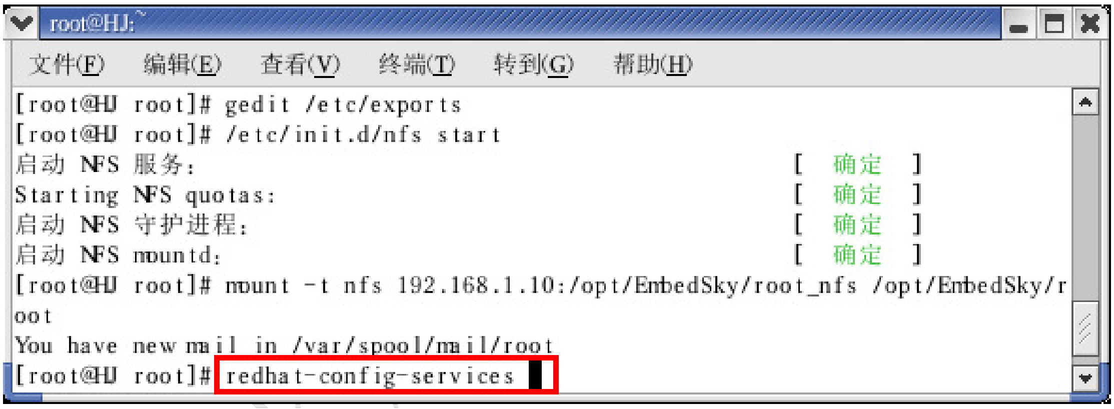
The contents under the directory "/opt/EmbedSky/root/" and "/opt/EmbedSky/root_nfs/" are the same. Operation on any one of the two will lead to change of the other. (caution: the directory "root" and "root_nfs" needs to be built by the user yourself. Use the command "mkdir xxx, xxx" to build a directory). The upper diagram indicates the user needs to input IP address of Linux on PC when mounting NFS.

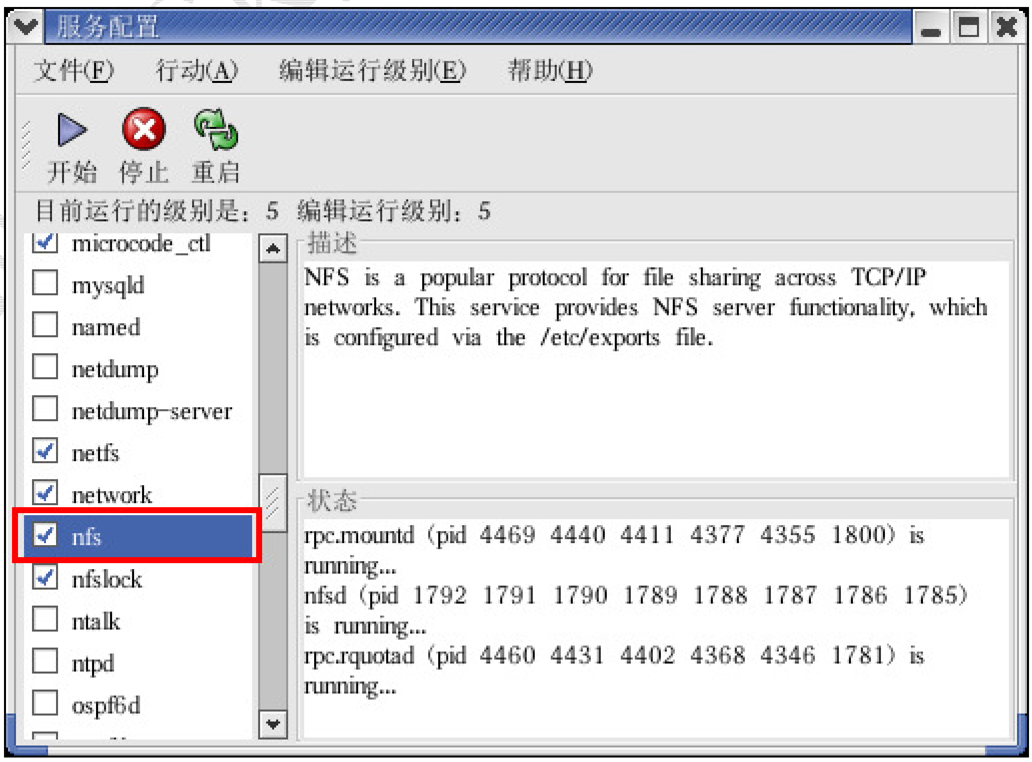


Use the command “#/etc/init.d/nfs stop” to stop NFS.

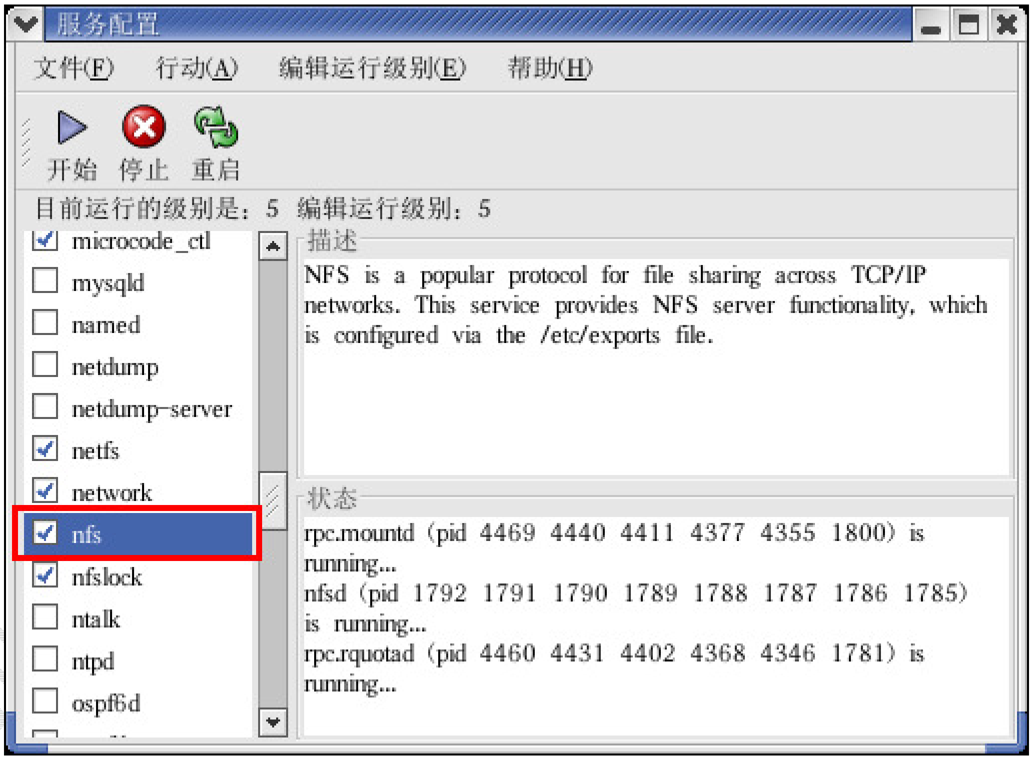
You can make NFS auto-start when PC starts up by following operation:

Use the command “#redhat-config-services” to open system service configuration window:





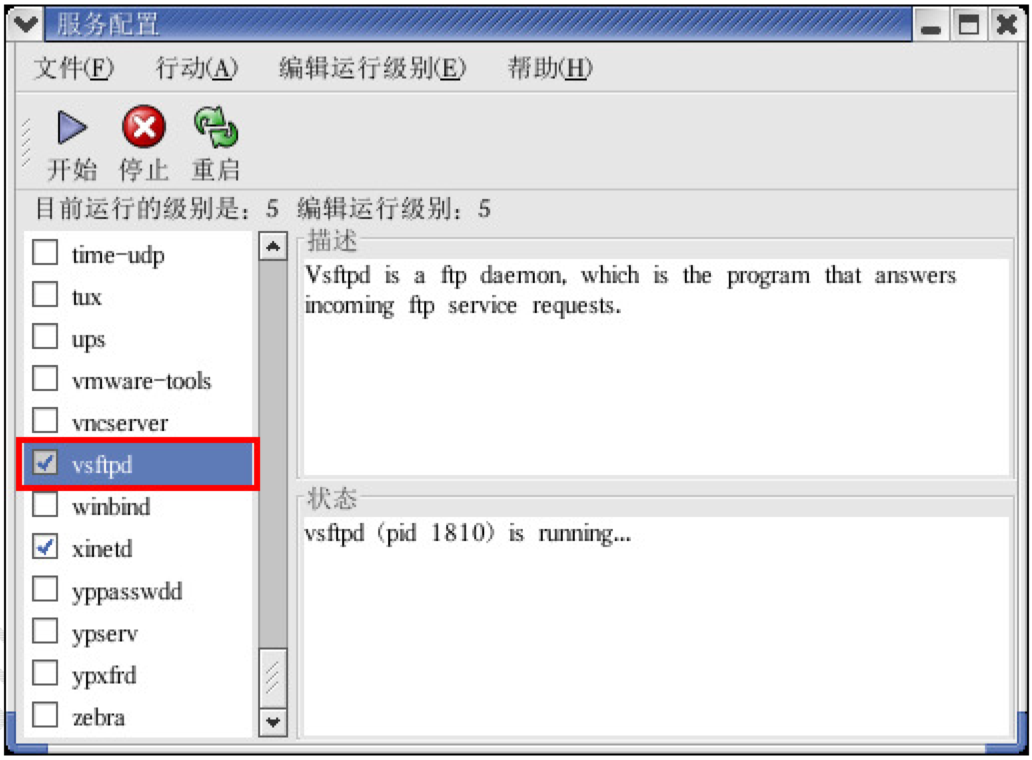
Find and select NFS option box:



Click “保存改变” in menu “文件(F)” to save configuration.

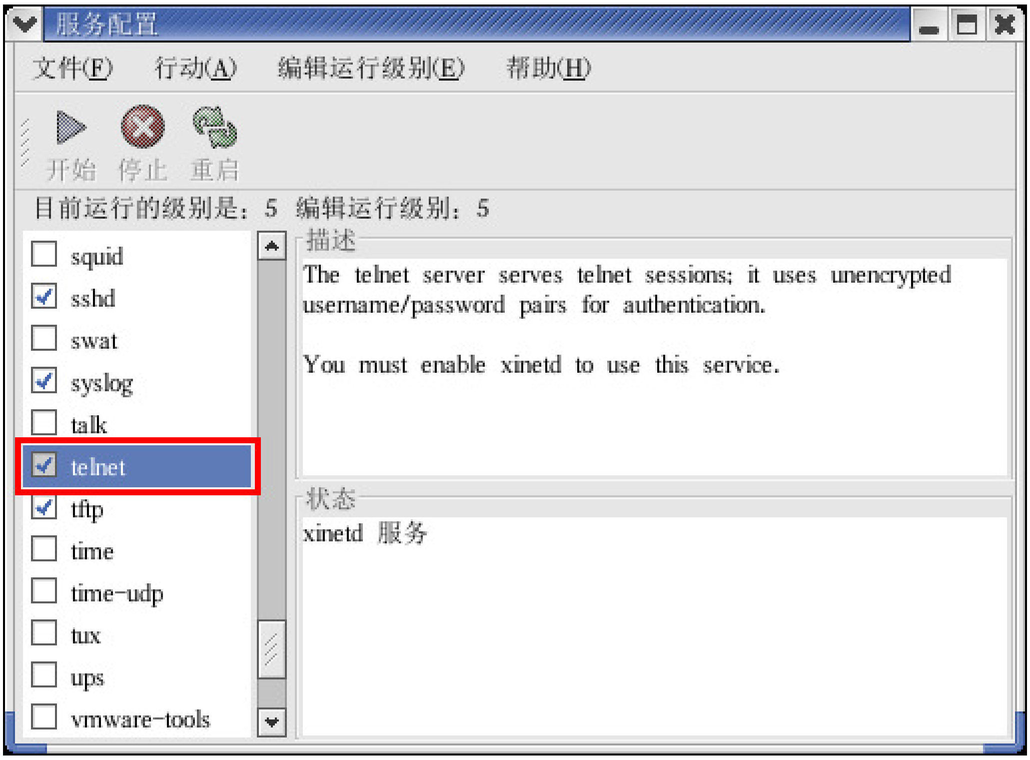
2. 5. 3 PC Linux FTP service configuration

As similar as NFS configuration, use the command “#redhat-config-services” to open system service configuration window and find the option “vsftpd”. Select it and save the configuration:



2. 5. 4 PC Linux Telnet service configuration

As similar as NFS configuration, use the command "#redhat-config-services" to open system service configuration window and find the option "telnet". Select it and save the configuration:

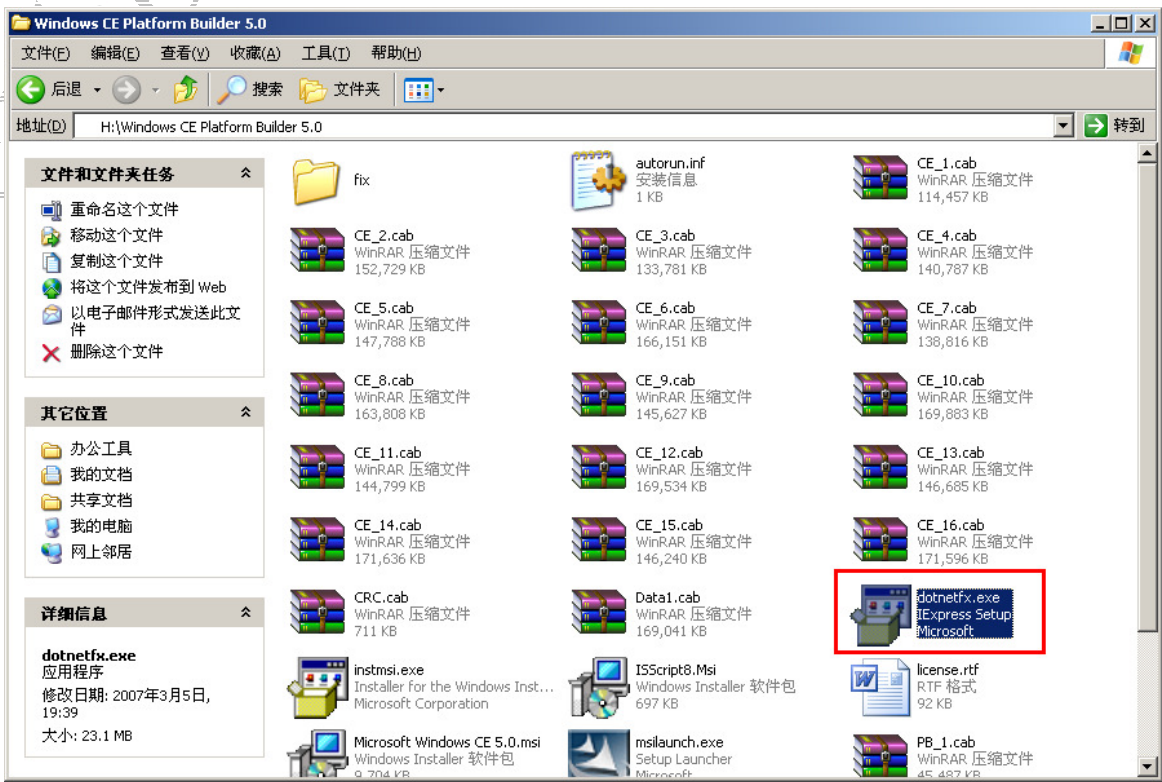


2. 6 WinCE environment configuration

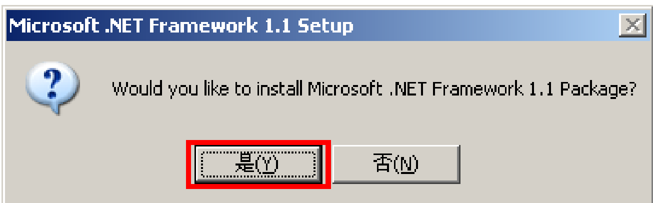
This section introduces the process installing Platform Build 5.0 (call it PB for short) in WindowsXP. We use the PB for developing and configuring WinCE kernel and debugging. A space larger than 5GB is needed by all PB files. In addition, an ordinary project needs about 500MB space. Therefore, it is suggested to provide a space no less than 7GB for PB installation.

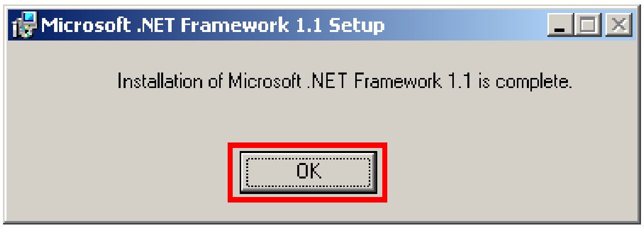
Caution: Copy the PB setup file to hard disk before installation. Otherwise the installing might fail.

Step 1: Open the WindowsCE CD, and install Framework net 1.0 first. Find “dotnetfx.exe” and double-click it to begin installation:

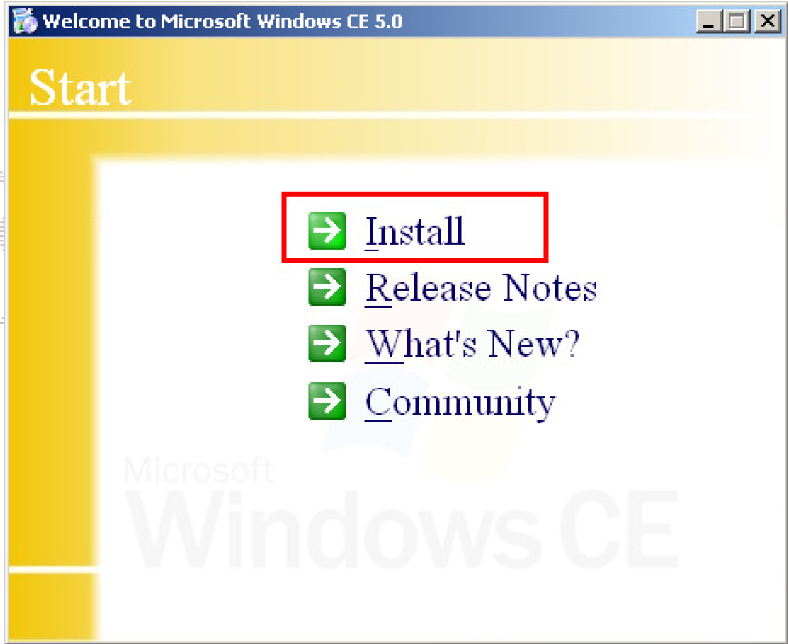


The following boxes pop up, keep clicking “是(Y)” button until the Framework net 1.0 installation finished.

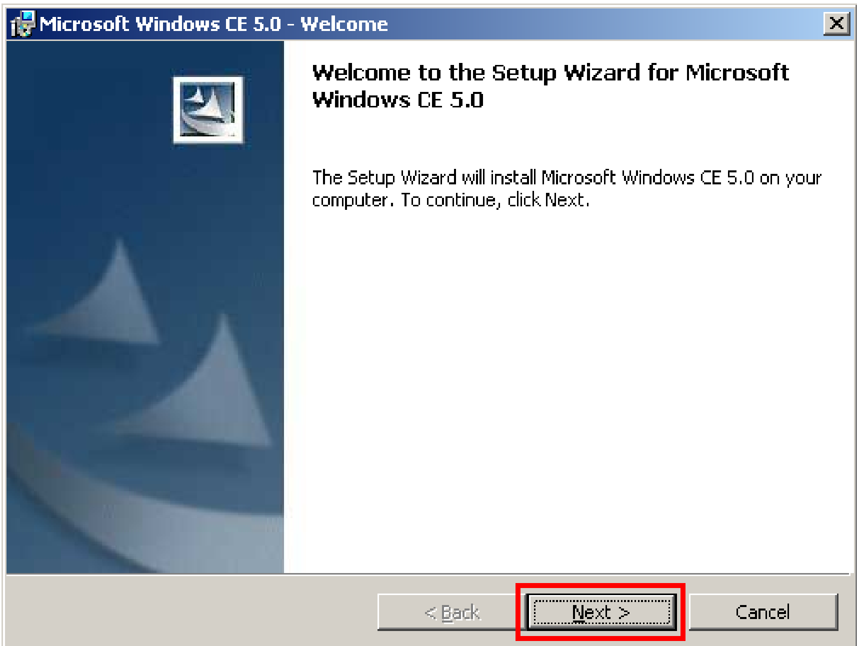




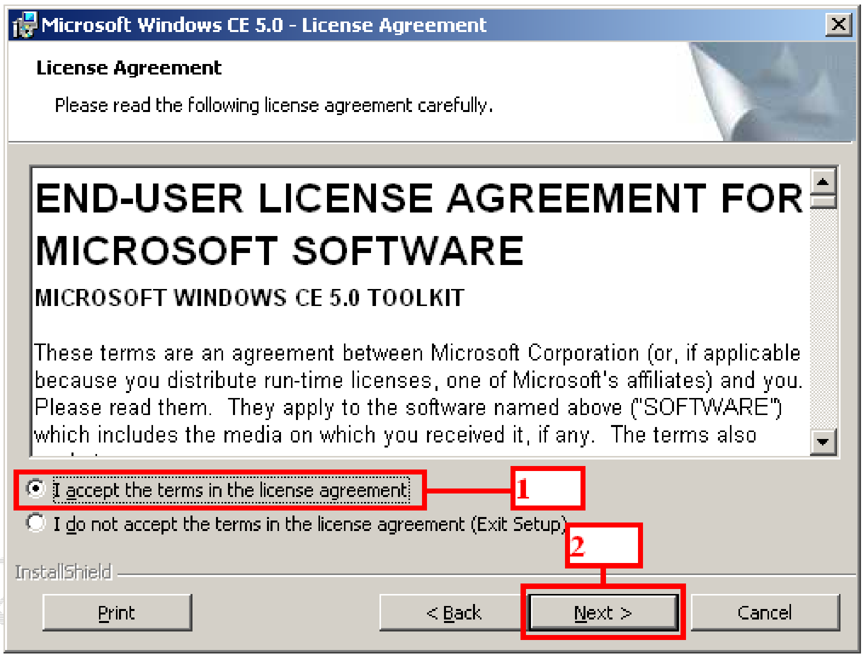
Step 2: Double-click “Setup.exe”. The following interface appears. Click “Install” to continue:



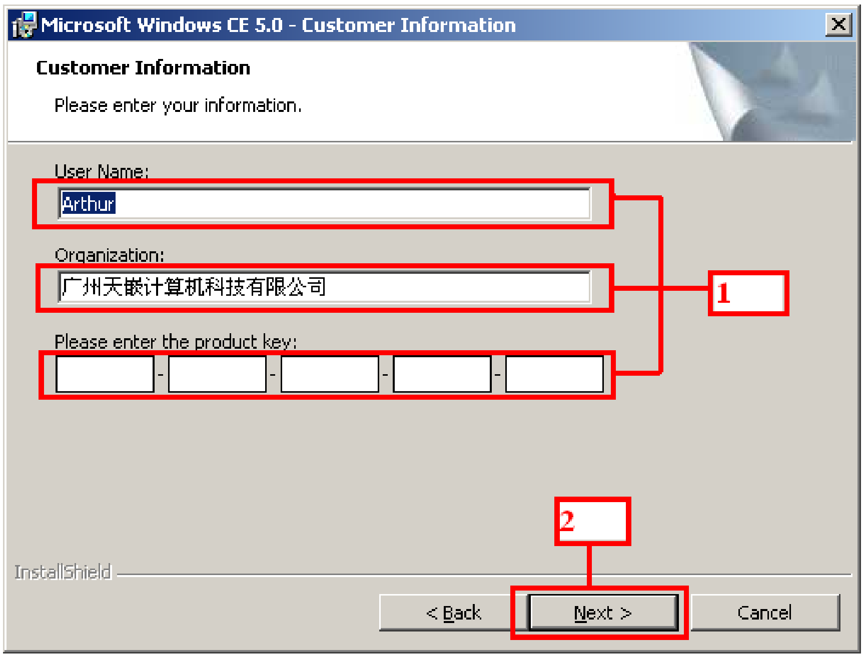
Step 3: The Welcome interface appears, click “Next” to continue:



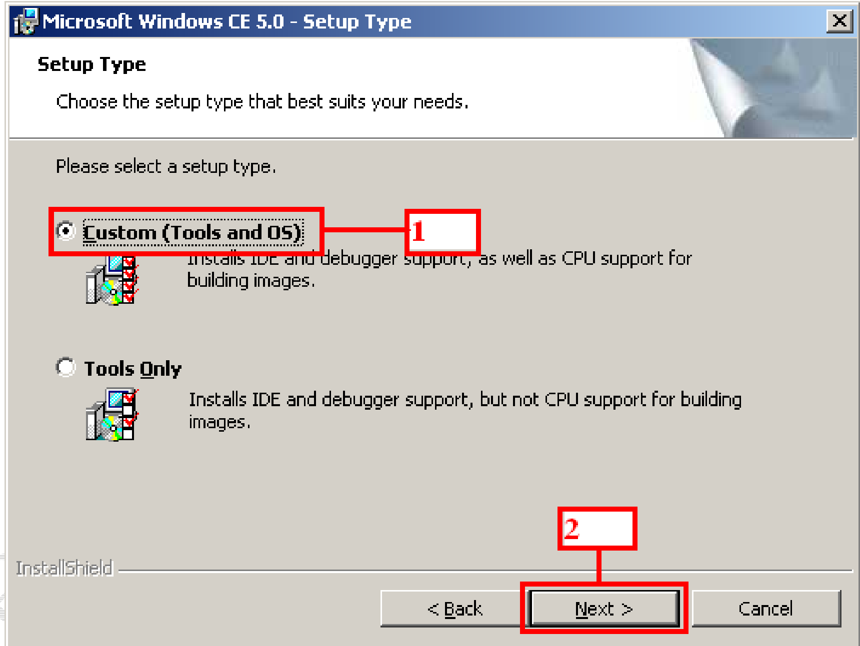
Step 4: The interface “License Agreement” appears, select the option “I accept the terms in the license agreement” and click “Next” to continue:



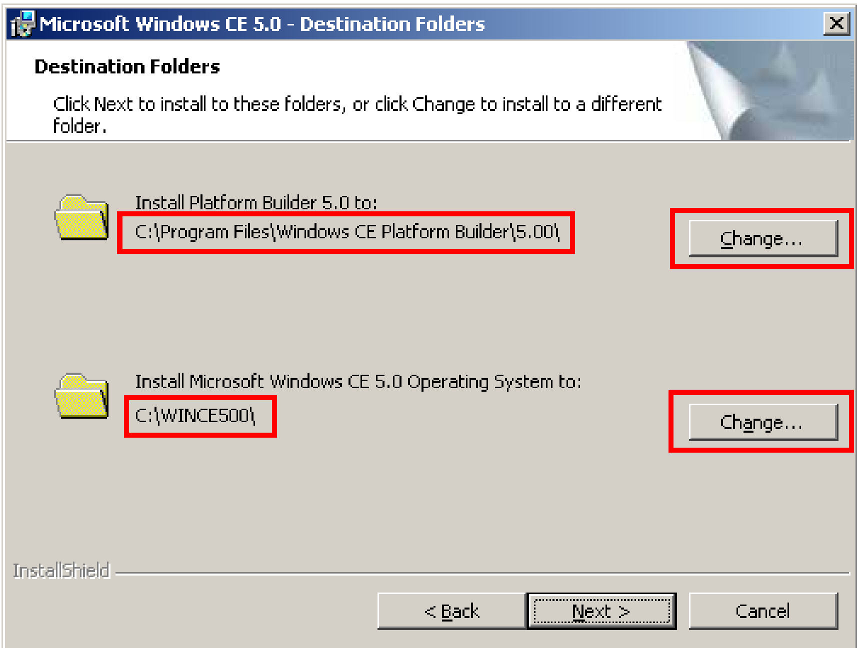
Step 5: The interface "Customer Information", enter the correct product key and click "Next" to continue:

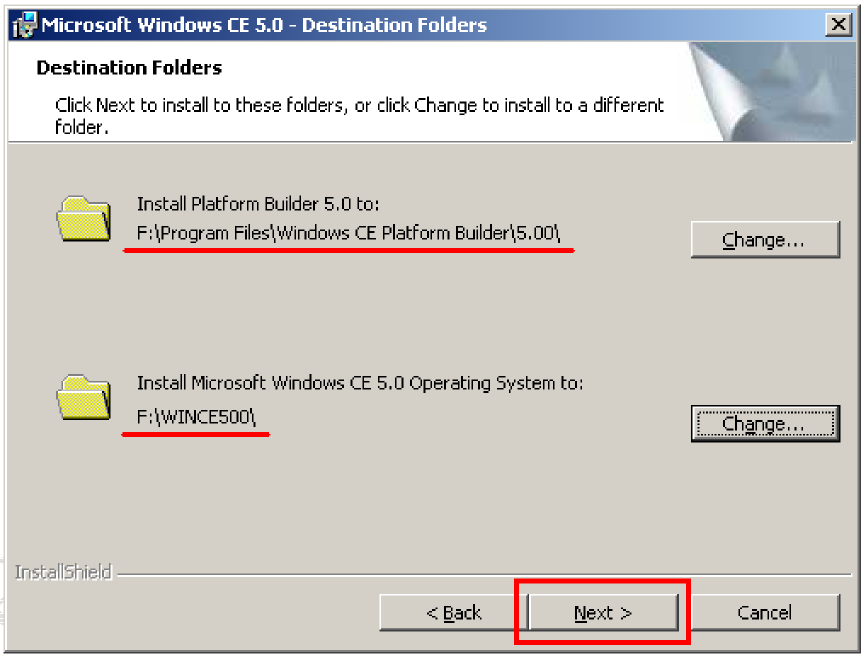


Step 6: Select a setup type. Select "Custom (Tools and OS)" and click "Next" to continue:

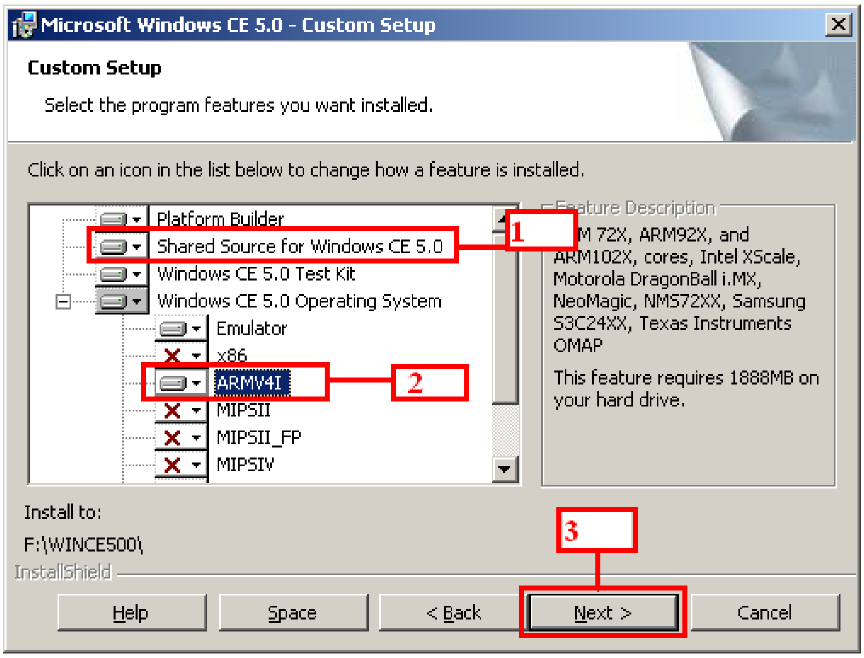


Step 7: Select the installation directory. The installation path here is F disk (The default directory is recommended). Click “Next” to continue: (The following two diagrams give two choices: choosing default C disk or choosing user-defined directory F disk)

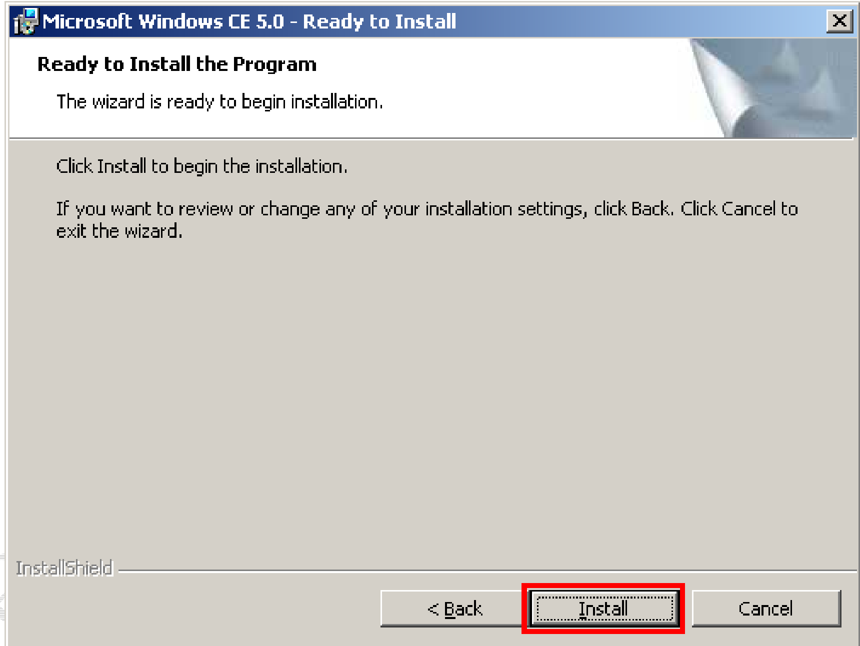




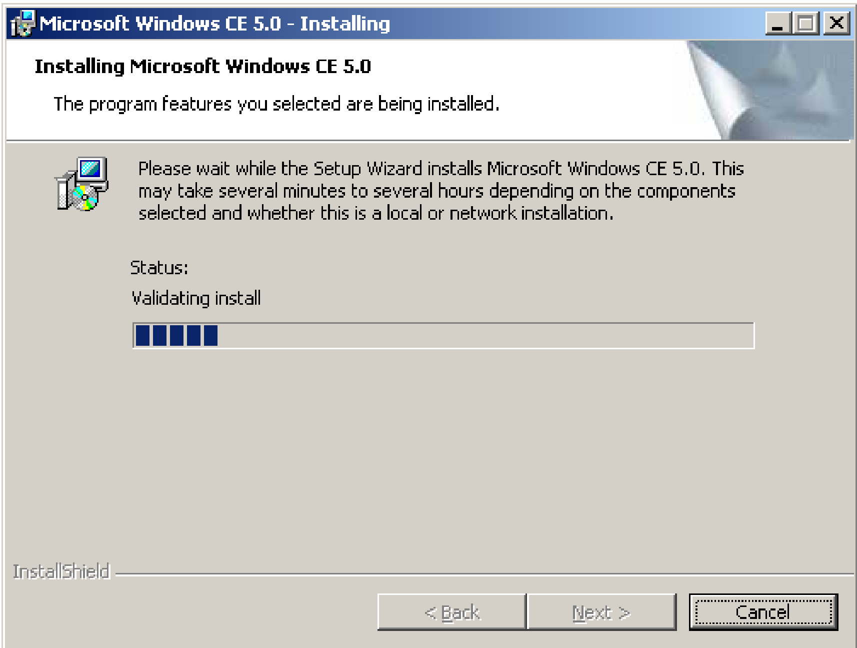
Step 8: The option "Shared Source for Windows CE 5.0" in the following diagram doesn't need to be selected. It is selected here for some reason of screen capture operation; When customizing your system platform, if you are a user of S3C2440, please select "ARMV4I" and continue:



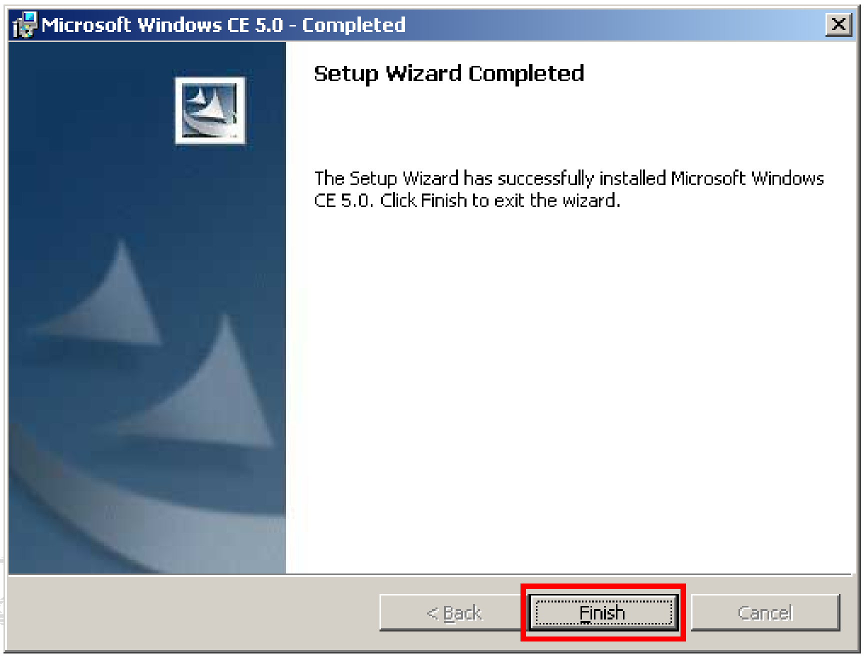
Step 9: confirm the installation. Click "Install" to continue:



Step 10: Installation starts:



Step 11: About 20 minutes later, PB installation complete. Click “Finish” to continue:



The process of PB installation is complete.

Chapter 3　Platform Utilization

Linux OS and Qt graphic interface (the buring files are u-boot_T35.bin, zImage_T35.bin and root_qt_tp.yaffs under the directory of "Images->Linux") has been installed in Development Board by default. We can also change the operating system into WinCE according to your request.

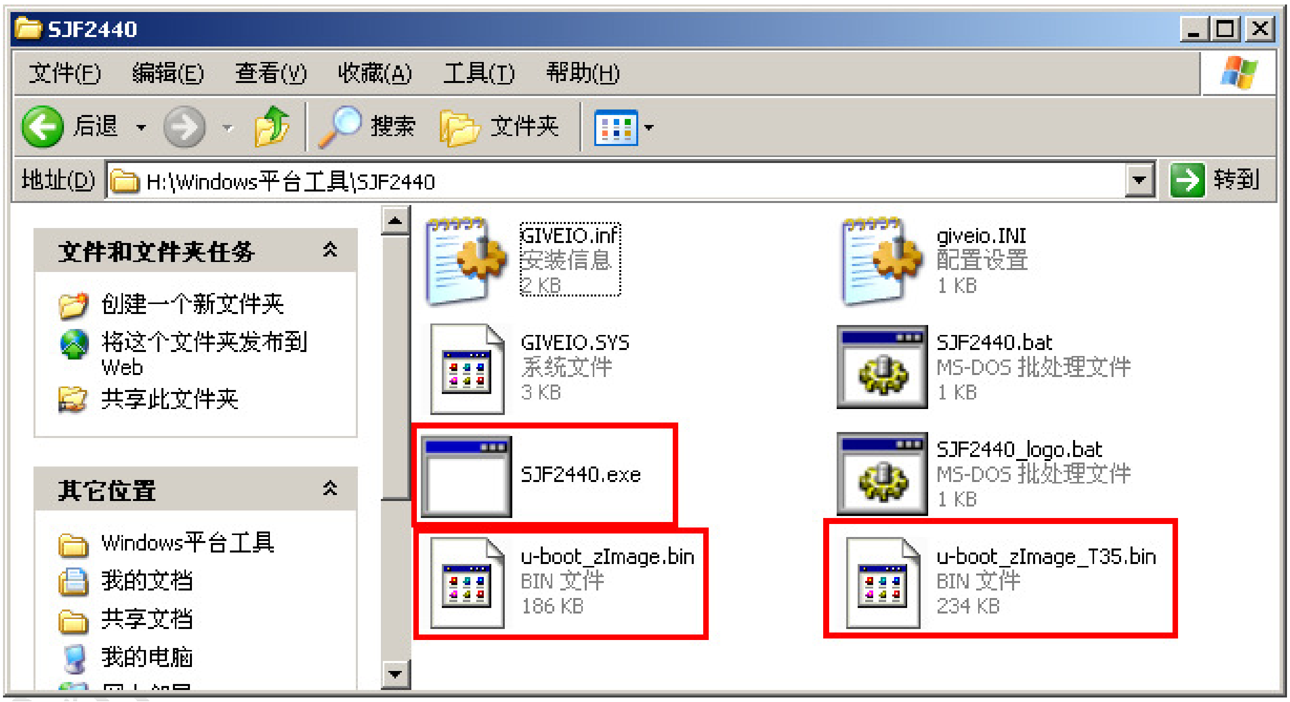
3. 1 Introduction of wire connection on platform and PC

Please connect the wire following the steps introduced in this section

- The jumpers: According to the default connection;
- Connect 5V power adaptor to power interface in platform (power adaptor output should be less than 7V);
- Audio interface: Green interface for output and the pink one for MIC input;
- Connect platform and PC via USB_Device interface
- Connect COM1 of platform and PC serial port with direct-connect serial port wire;
- Connect 100M network card of platform and PC with net line;
- Connect Jtag interface of platform and PC parallel port with Jtag download board;
- Connect camer module to camera interface of platform;
- Connect LCD module to LCD interface of platform with FFC (Flexible Flat Cable);

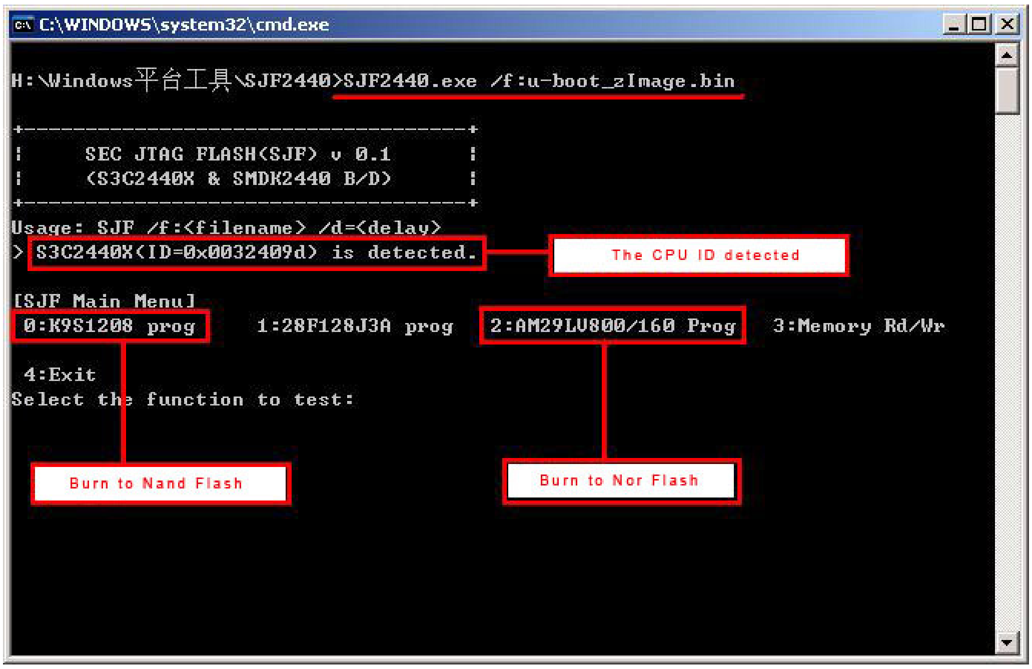
3. 2 Burning u-boot by SJF2440

The software SJF2440 is under the directory "SJF2440" of "Windows 平台工具" in CD-ROM. SJF2440.exe is used for burning software; u-boot_zImage.bin and u-boot_zImage_T35.bin are u-boot image files; SJF2440.bat and SJF2440_logo.bat are batch processing files for burning:



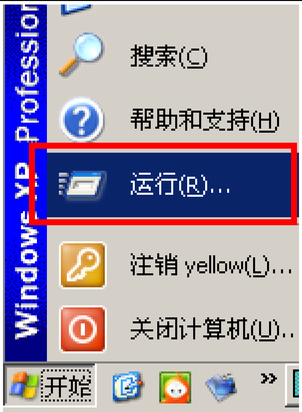
3. 2. 1 Executing SJF2440 with batch-processing files

Double-click "SJF2440.bat" in the upper diagram:

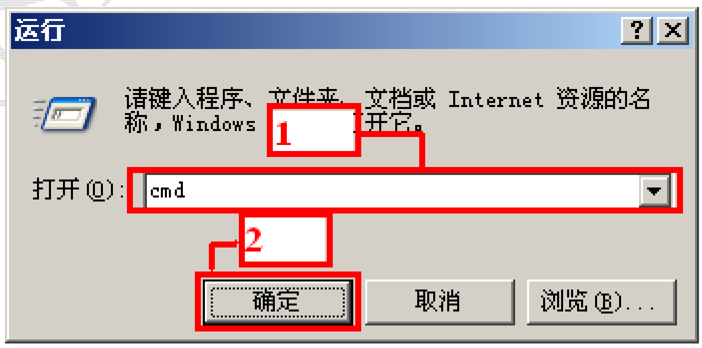


3. 2. 2 Running SJF2440

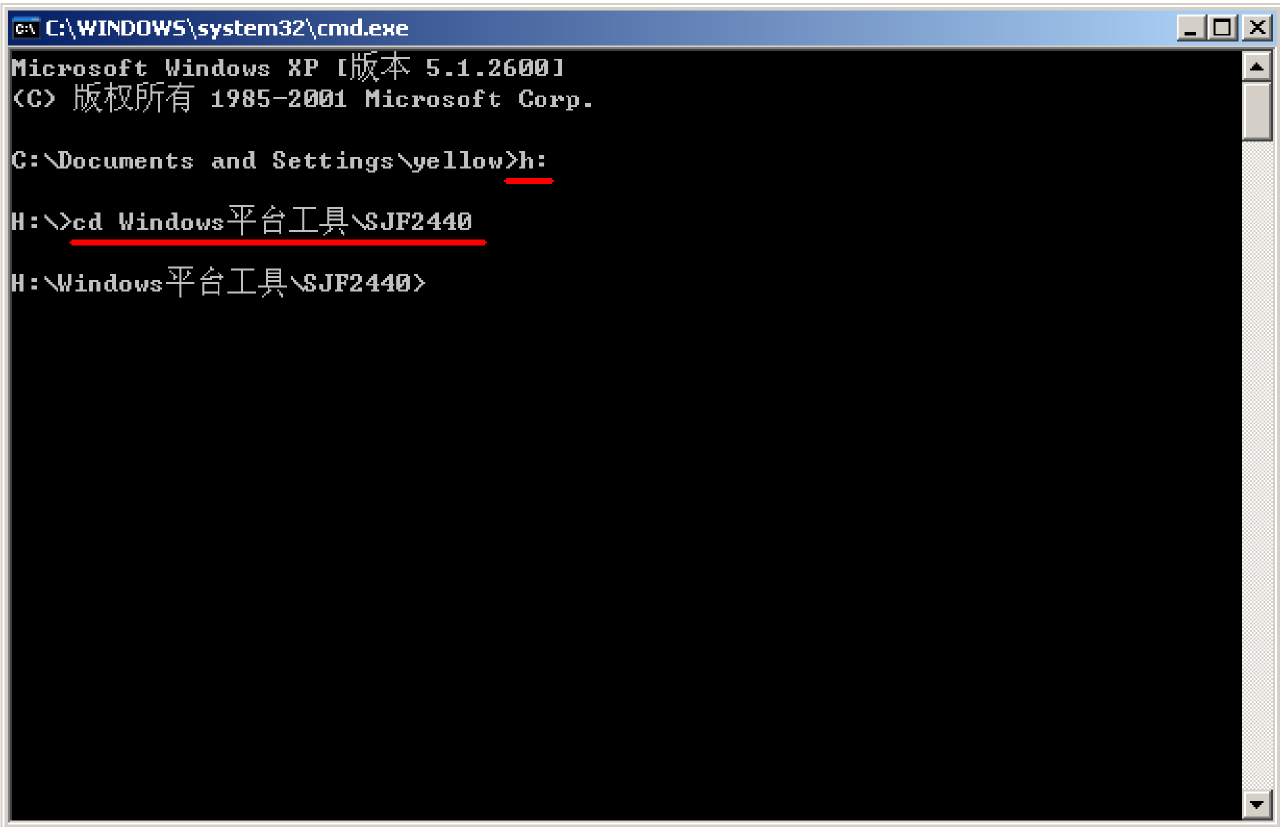
Step1, click "运行" in menu "开始":



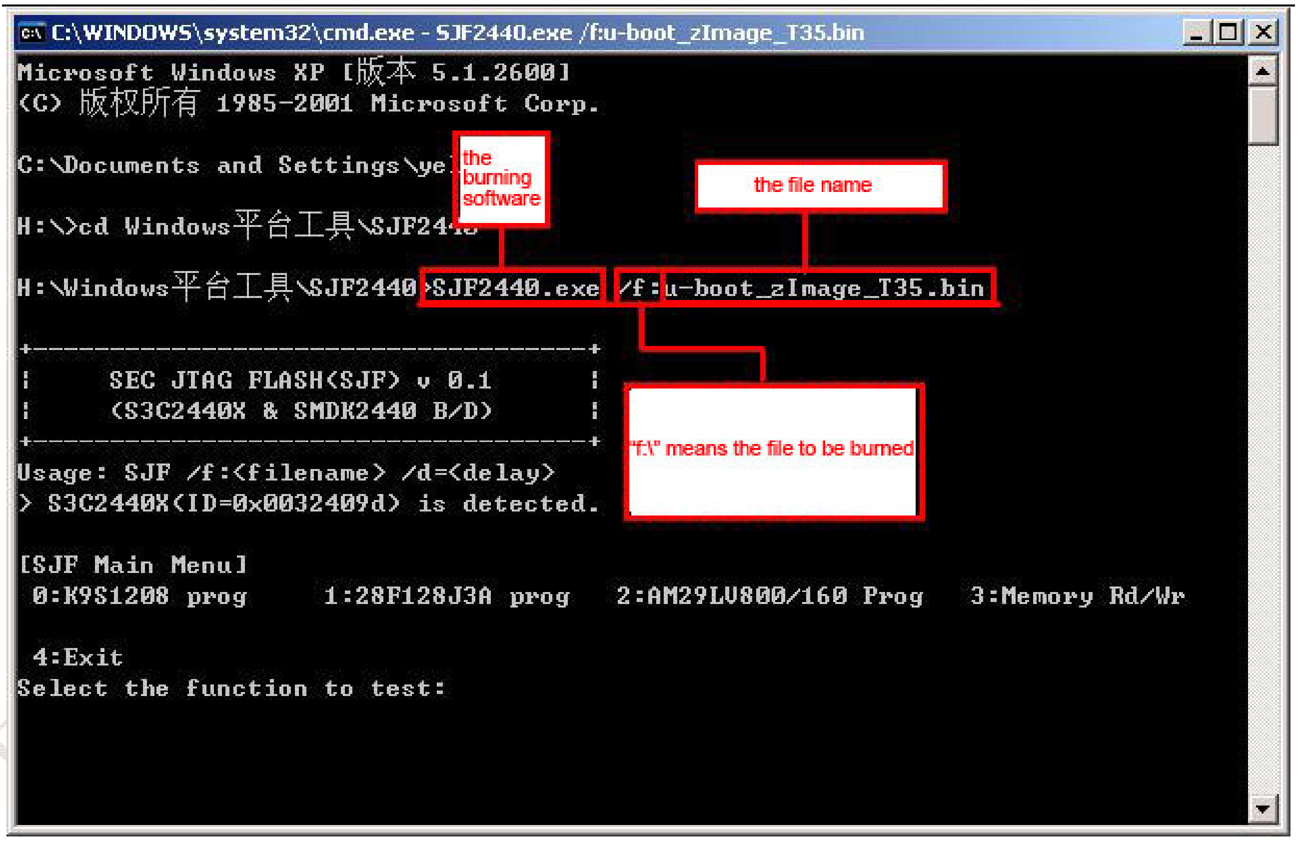
Step2, enter “cmd” and click “确定” to continue:



Step3, execute DOS command to enter into SJF2440 directory shown in the following diagram:



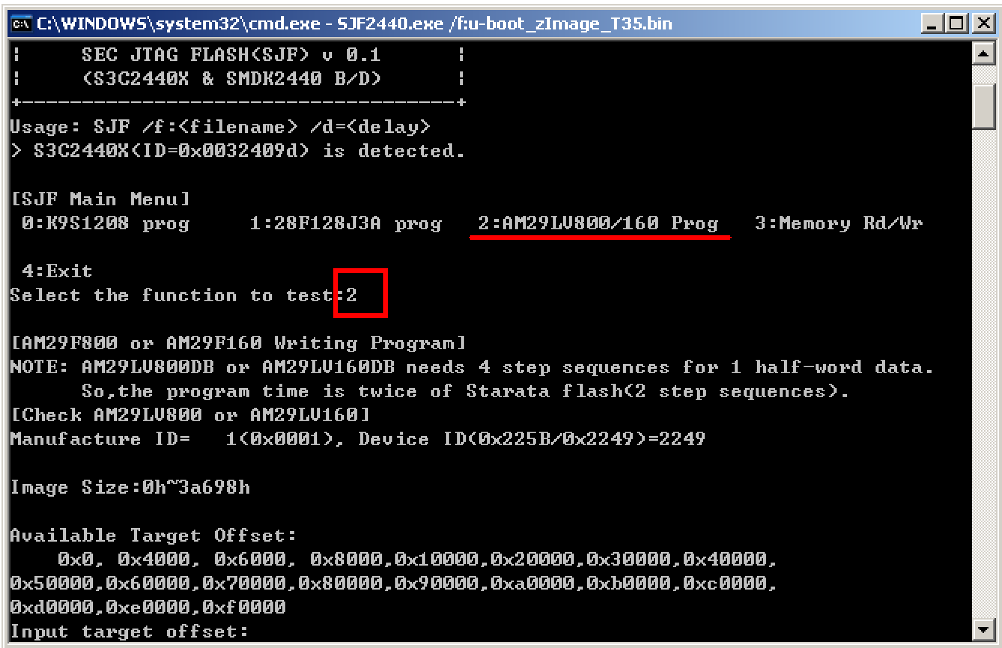
Step4, run SJF2440. execute the following command: SJF2440.exe /f:u-boot_zImage_T35.bin (instruction: /f: dosen’t mean disc but file), as shown in the following diagram:如下图所示：



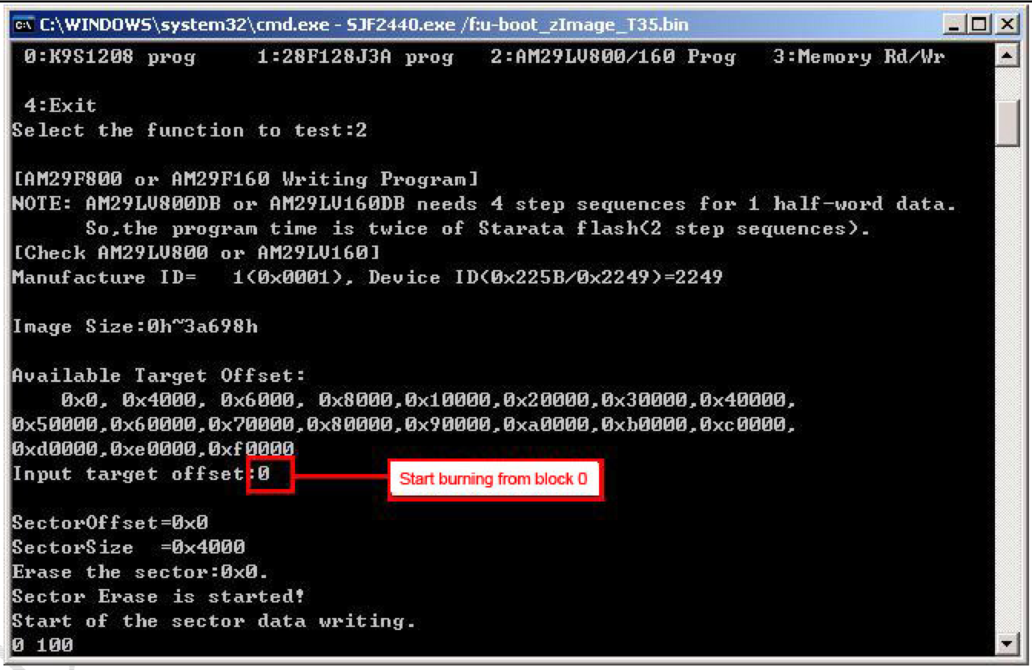
3. 2. 3 Burning u-boot to Nor Flash

The burning process:

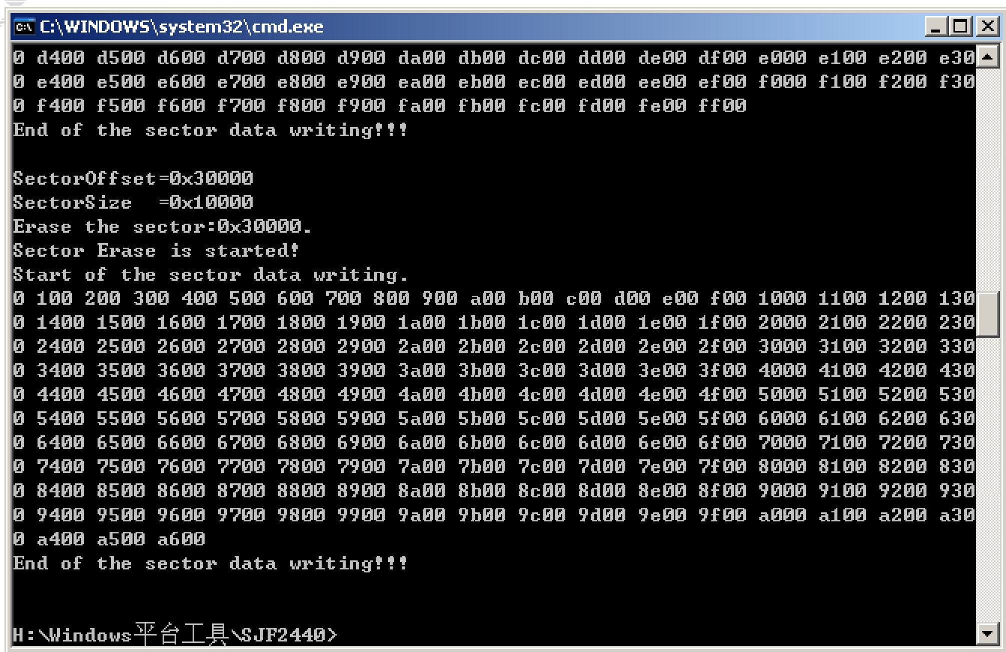
Step1, in previous diagrams of “3.2.1 节”, select “2” and press return key to continue:



Step2, enter “0”. Start burning from block 0 and press return key to continue:



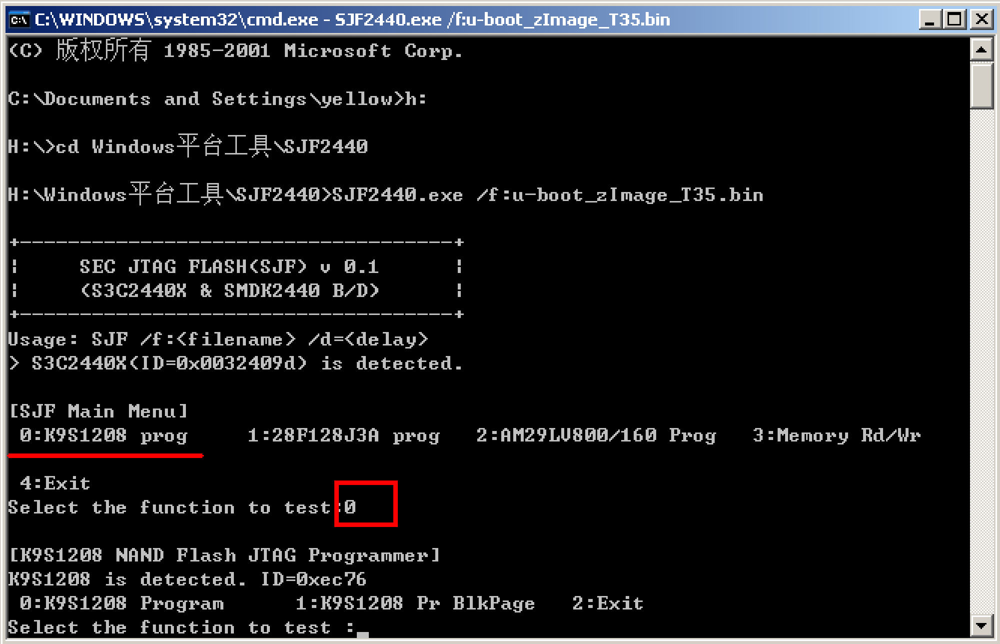
Step3, when burning process is finished, it exits automatically or the following diagram appears:



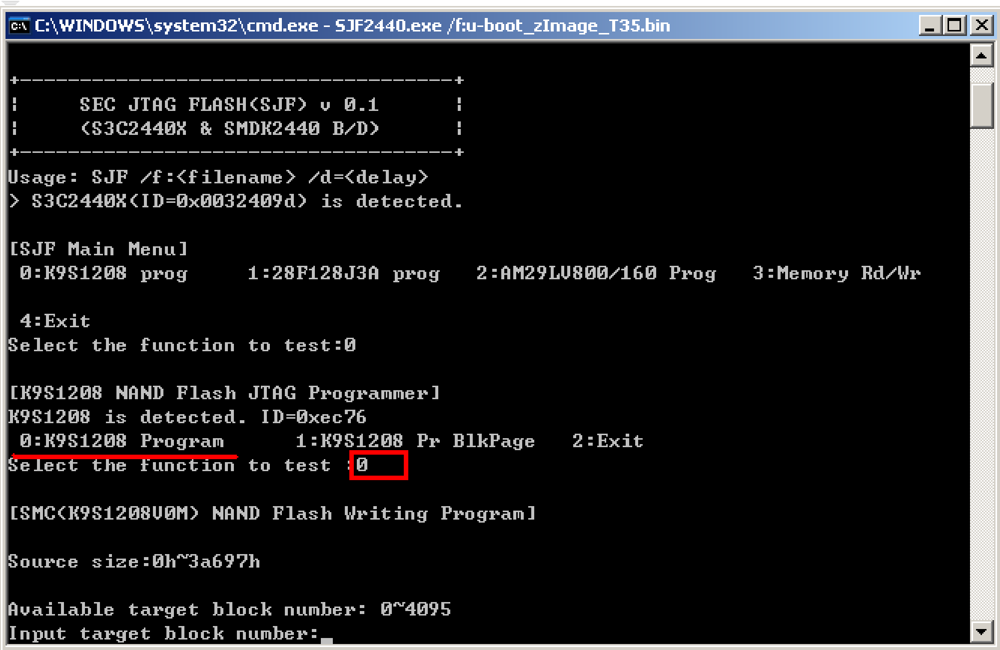
3. 2. 4 Burning u-boot to Nand Flash

The burning process:

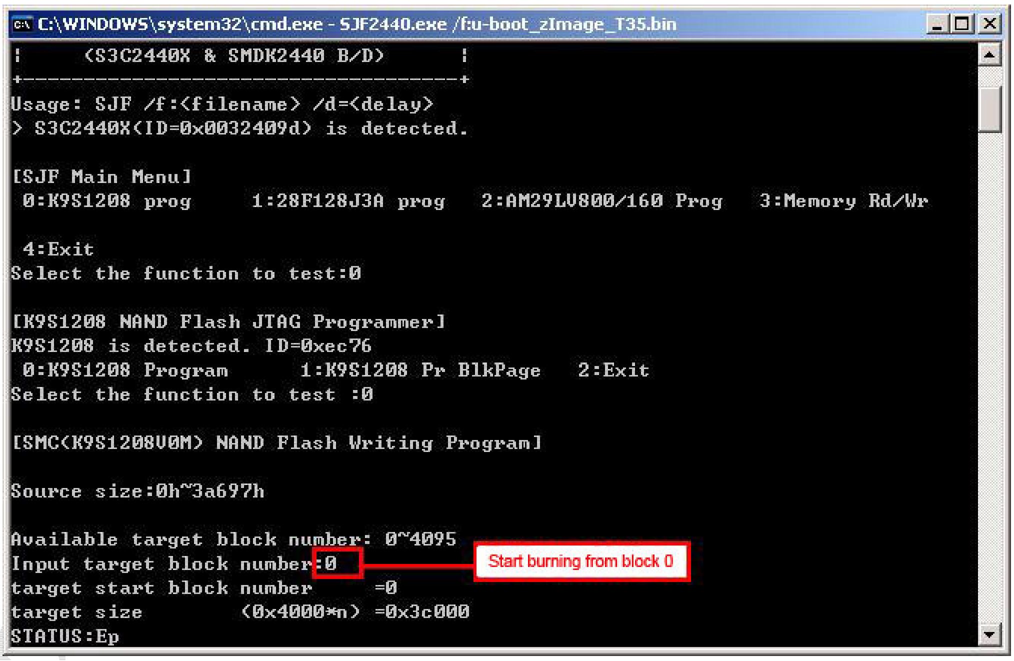
Step1、in previous diagrams of “3.2.1 节”, select “0” and press return key to continue:



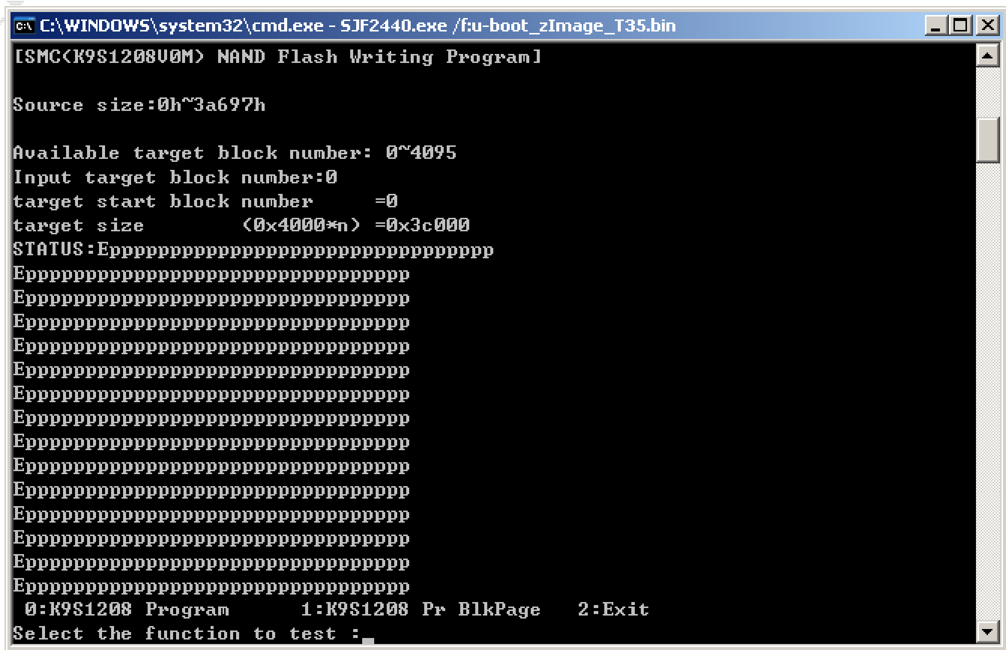
Step2, enter “0” to select Flash type and press return key to continue:



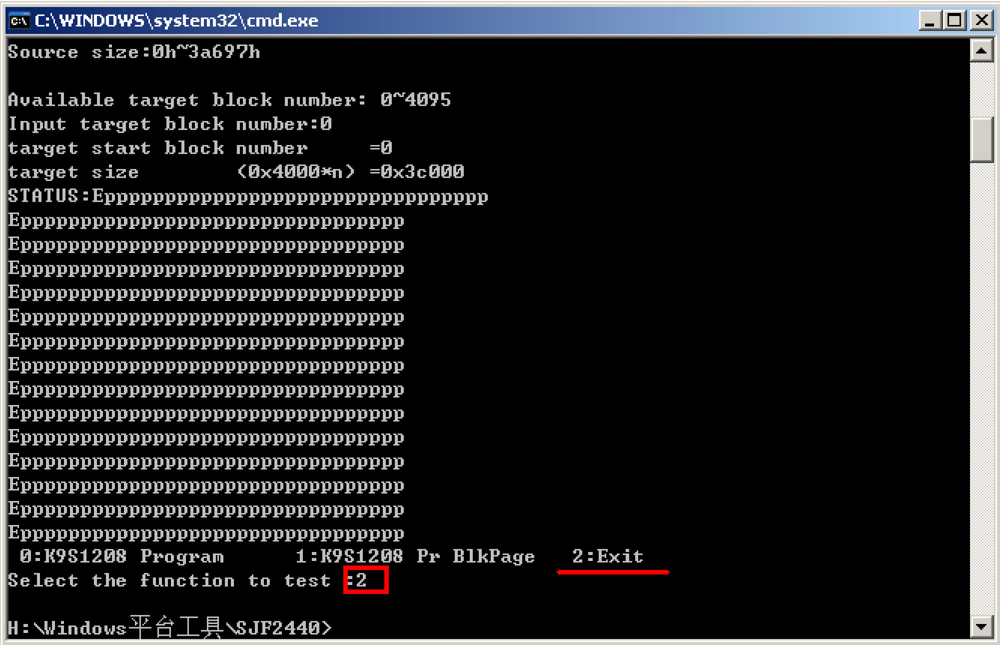
Step3, enter “0”. Start burning from block 0 and press return key to continue:



Step4, when burning process is finished, the following diagram appears:



Select “0” to continue or “2” to exit. Here we select “2”:



3.3 Burning u-boot with H-Jtag

3.3.1 H-JTAG installation

Decompress and install “H-JTAG V0.4.3.zip” under the directory “H-JTAG” of “Windows 平台工具” in CD-ROM; Or download other versions from the website “http://www.hjtag.com/download.html”. The following instructions are corresponding to the version “H-JTAG V0.4.3.zip” in CD-ROM.

Step1, double-click “H-JTAG V0.4.3.zip” to decompress it. If you have no decompression tool, please install it first;

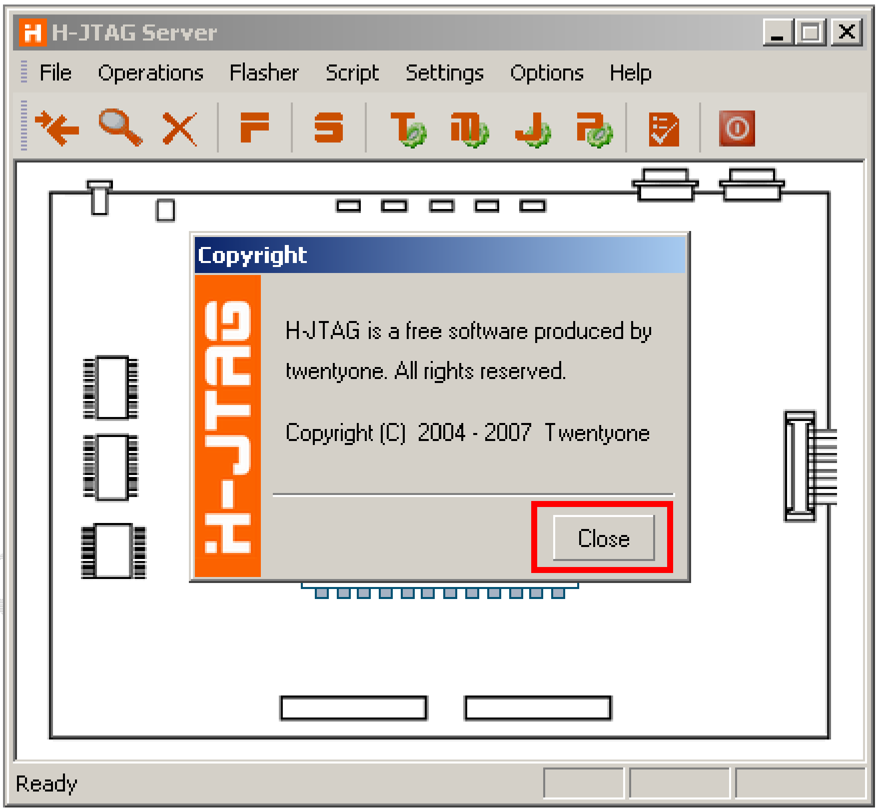
Step2, H-JTAG V0.4.3.EXE appears after decompressing, double-click it to enter into installation guide. Keep clicking “next” in the following pop-up interfaces.

Step3, after installation, the shortcut “H-JTAG” and “H-Flasher” will appear in desktop.

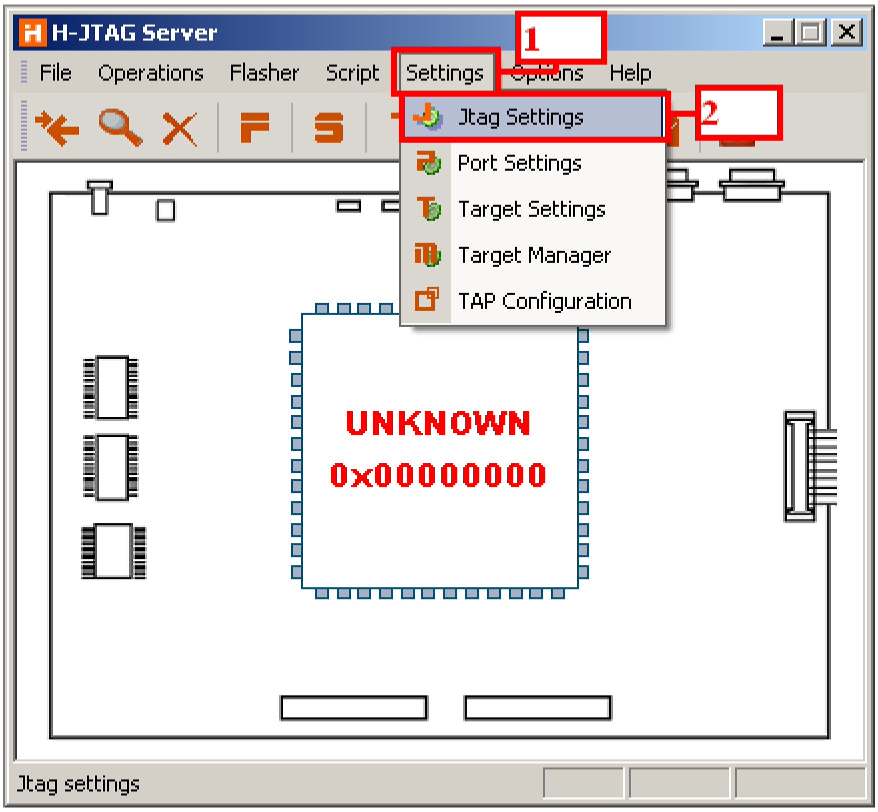
3.3.2 H-JTAG configuration

The following diagrams illustrate the configuration steps:

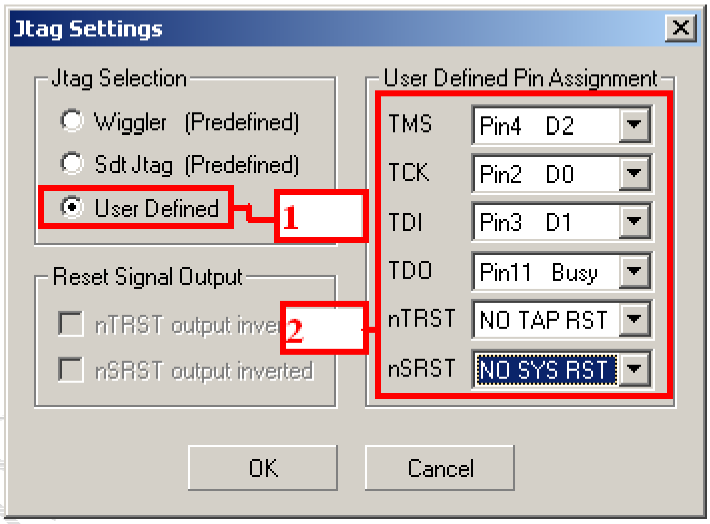
Step1, double-click the icon “H-JTAG” as the following diagram:



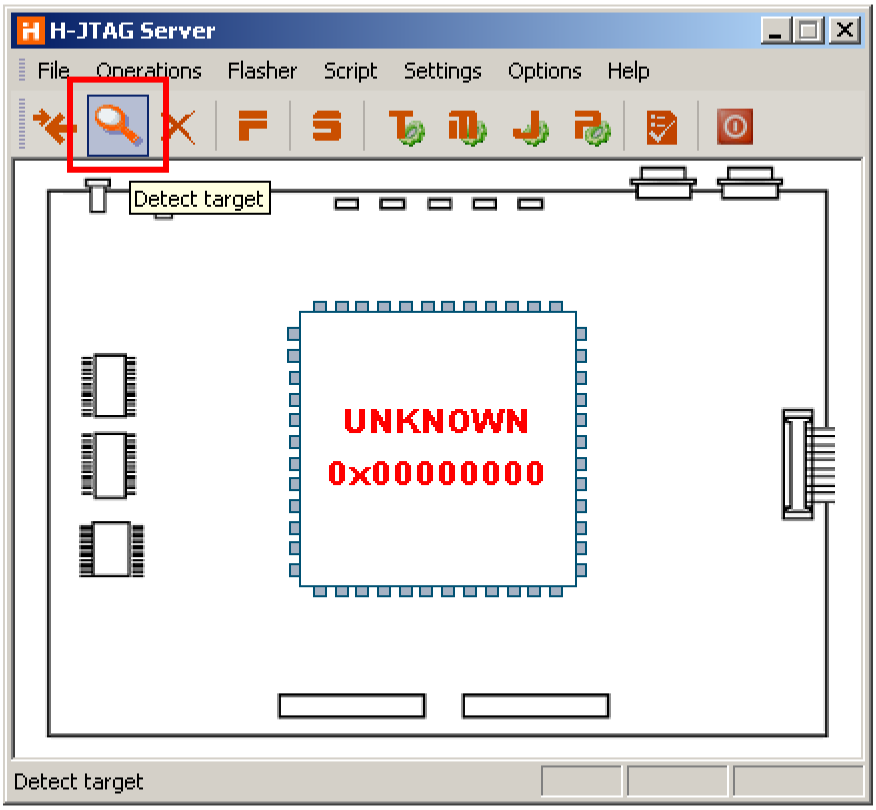
Step2, click “close”. The following interface appearing, click “Jtag Settings” in menu “Settings (设置)”:



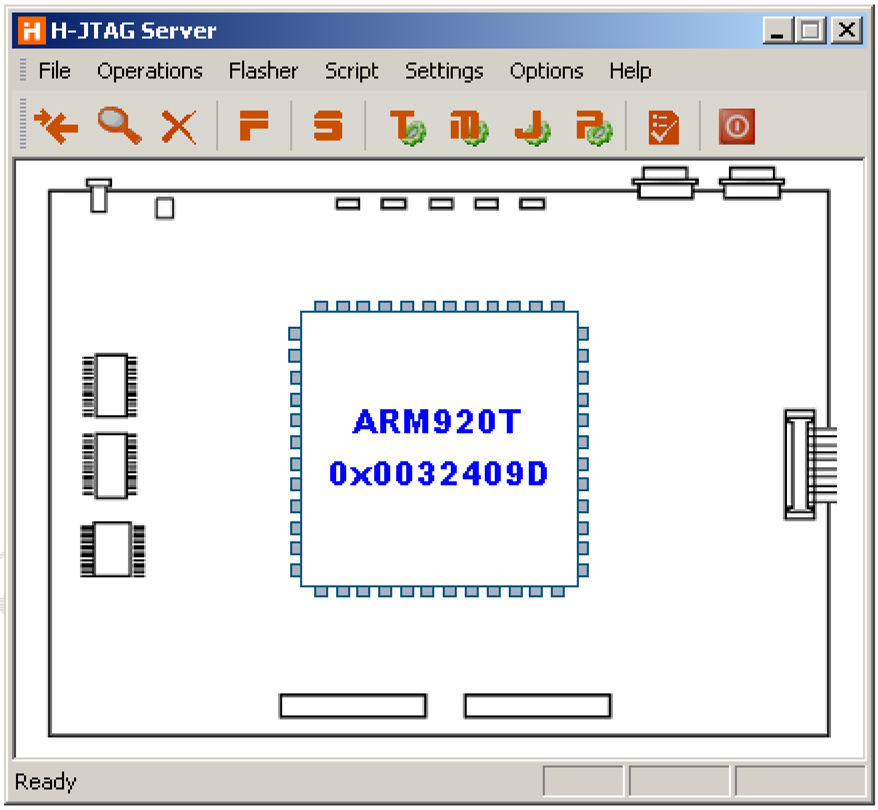
Configure the following interface “Jtag Settings”:



Step3, click “OK” to finish configuration. Check all the connections in Development Board and turn on the power. Click “Detect target”() or click “Detect target” in menu “Operations”:



Step4, if CPU has been detected, the following diagram appears. Or prompting error if failed.

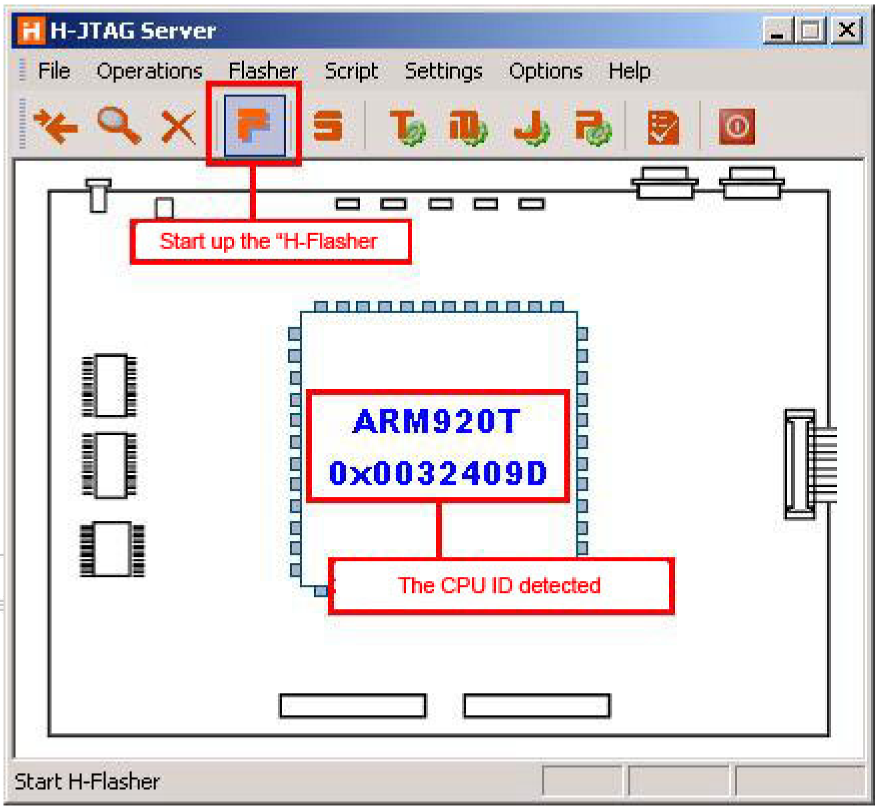


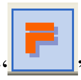
3. 3. 3 H-Flasher configuration

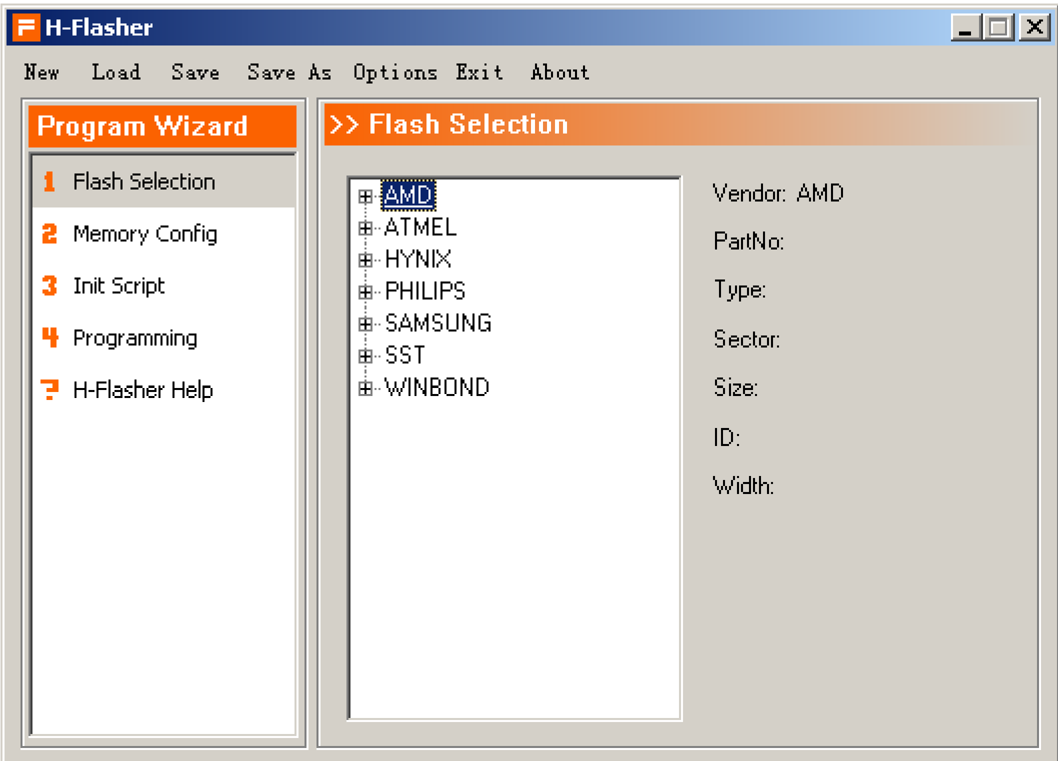
Caution: The software “H-Jtag” can only burn u-boot into Nor Flash.

Step1, connect Jtag wire and remove the jumper on OM0. Then turn on the power;

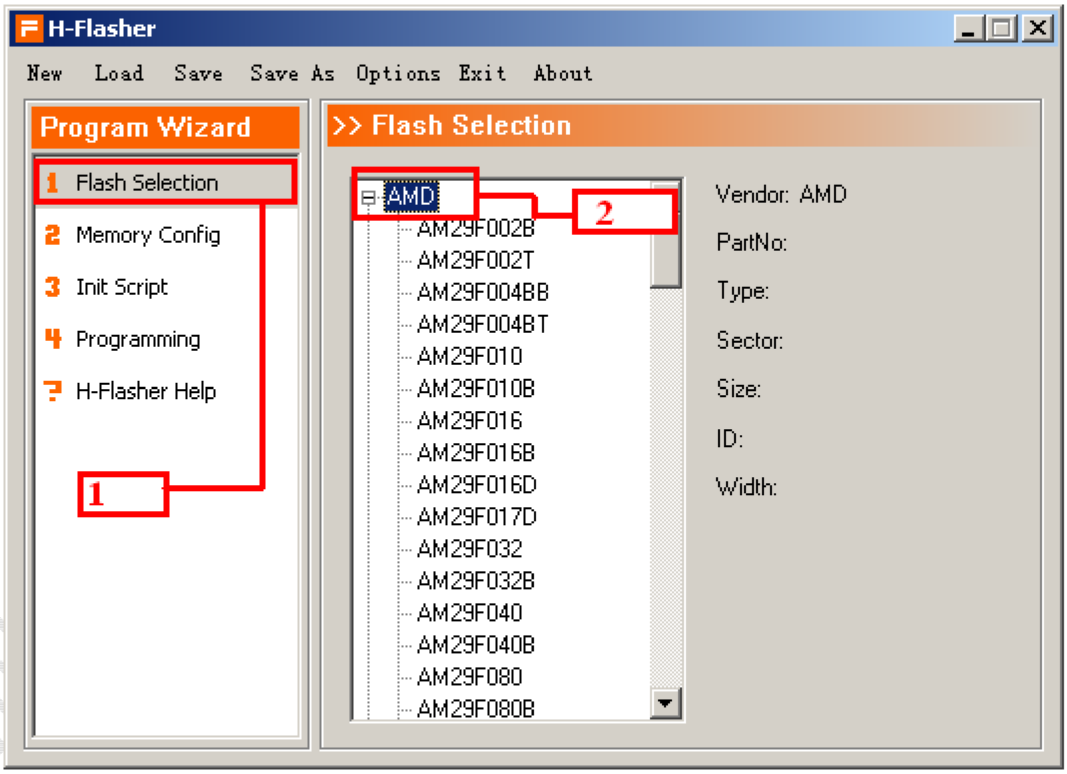
Step2, start up “H-Jtag”. The software begins to dectet CPU automatically:



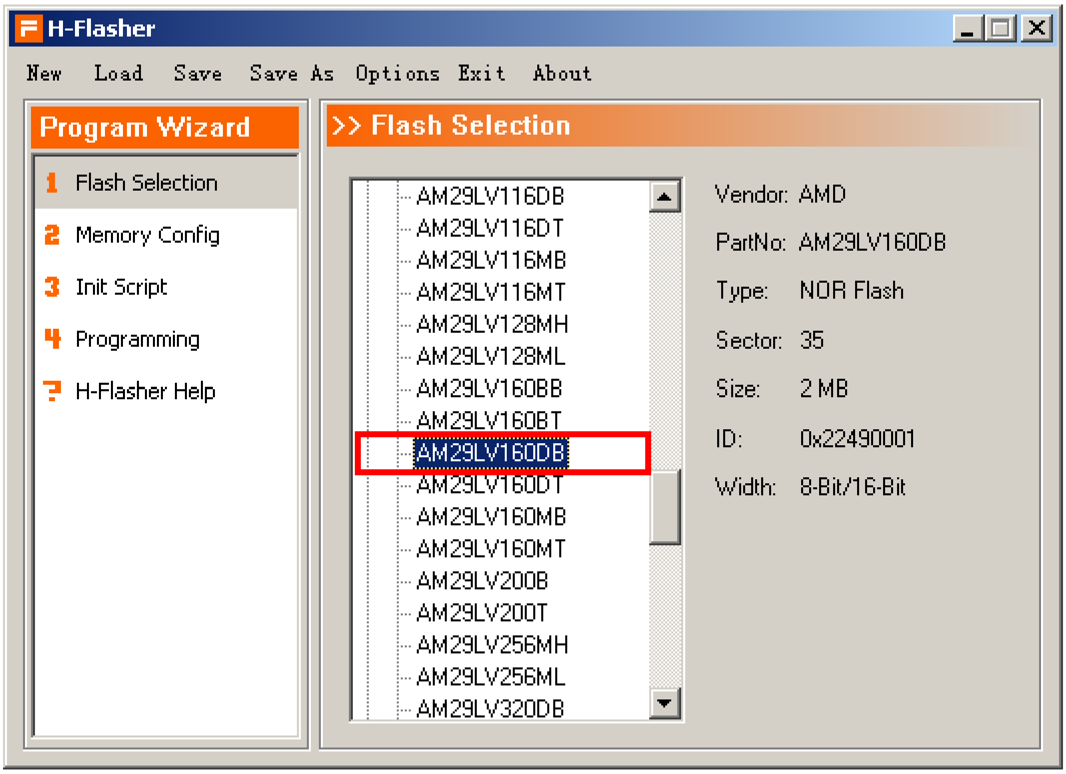
Step3, click the icon "" to start up software "H-Flasher":



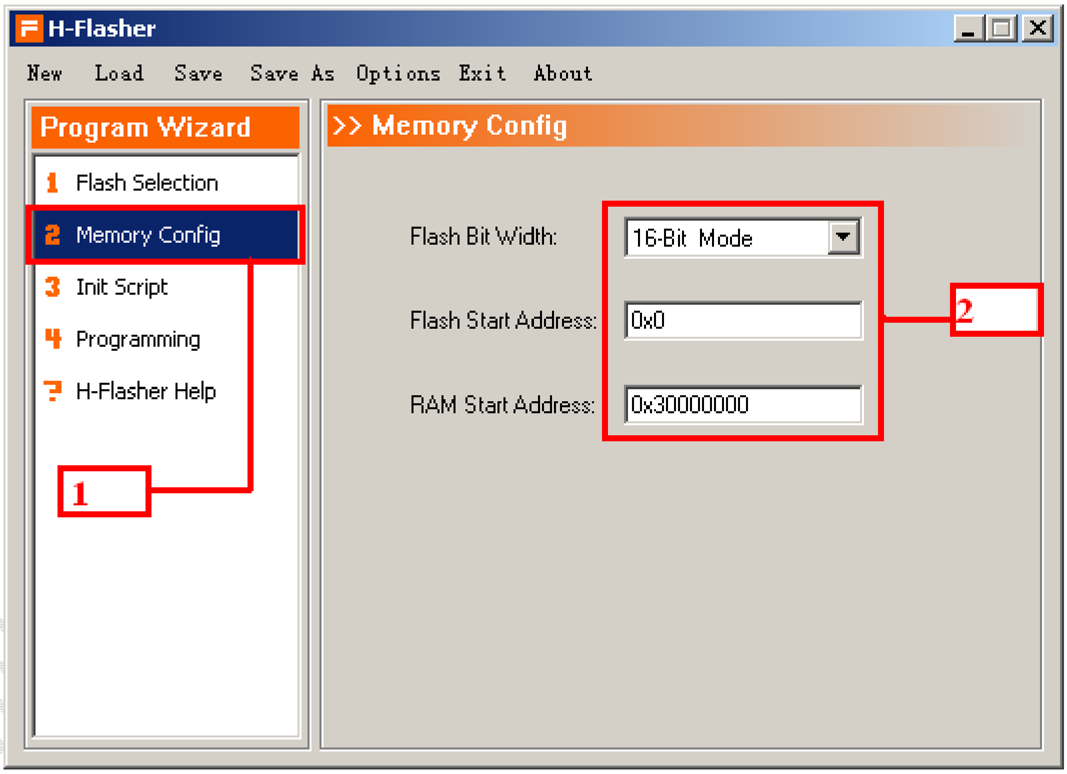
Step4, select "1 Flash Selection" and click "AMD" on the right:



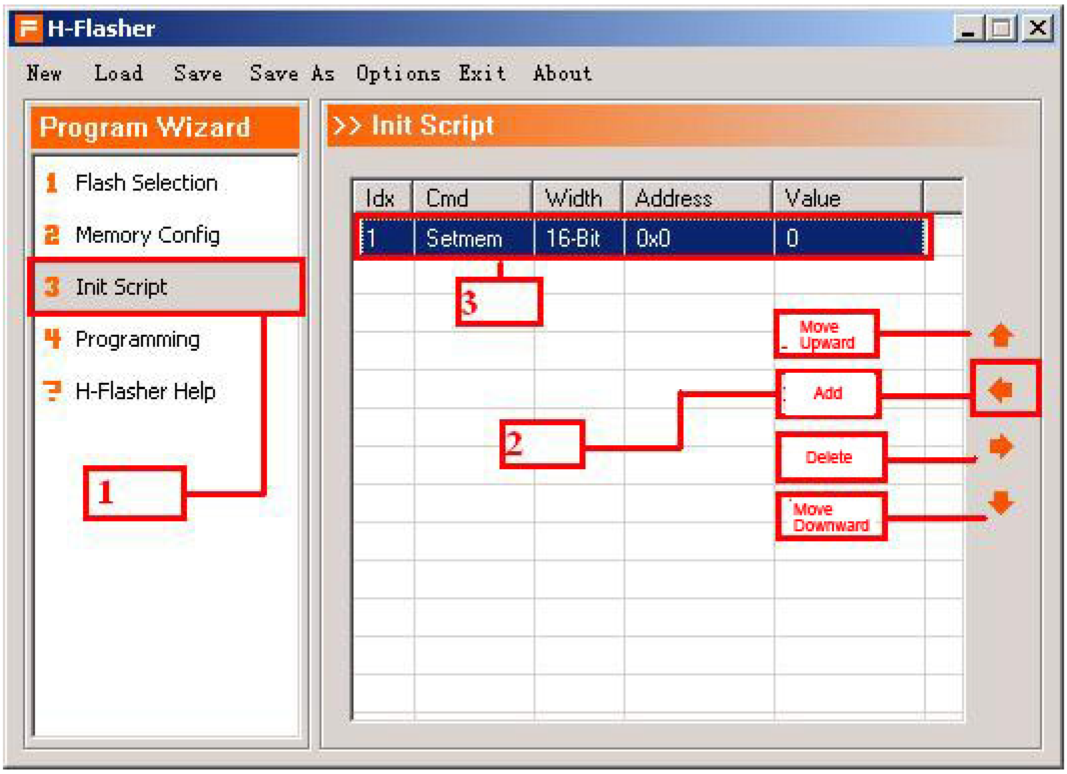
Find “AM29LV160DB” downwards and click it:



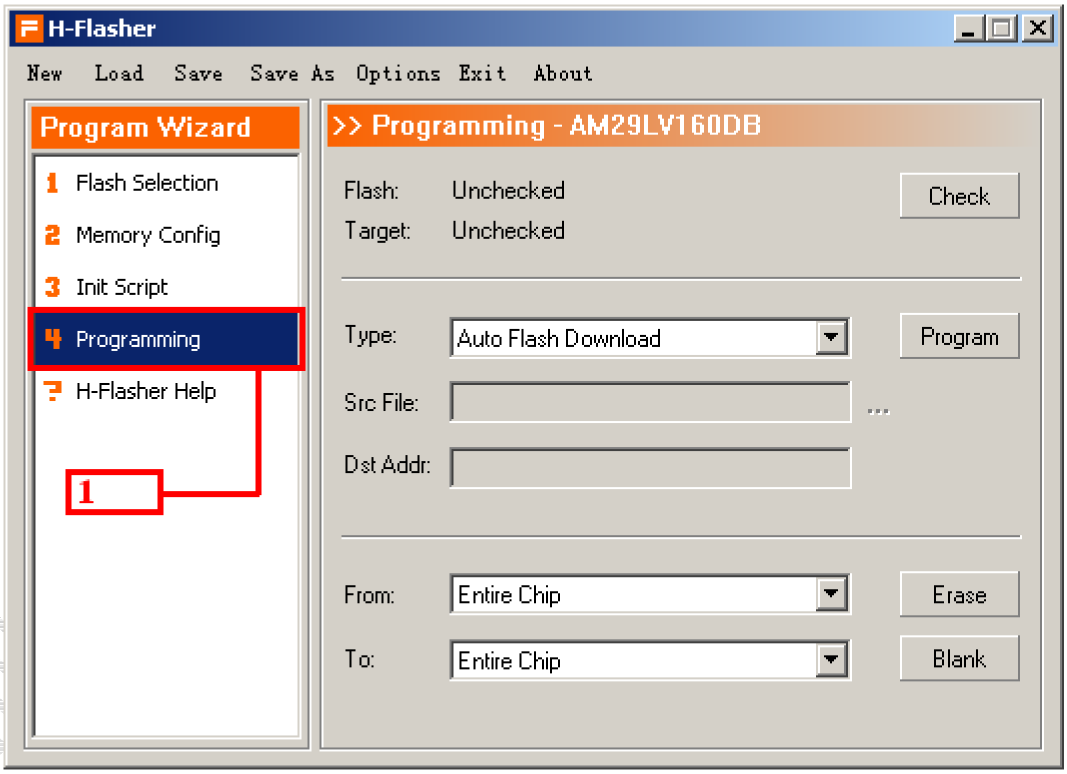
Step5, select “2 Memory Config” and configure the options rightward. As shown in the following diagram:



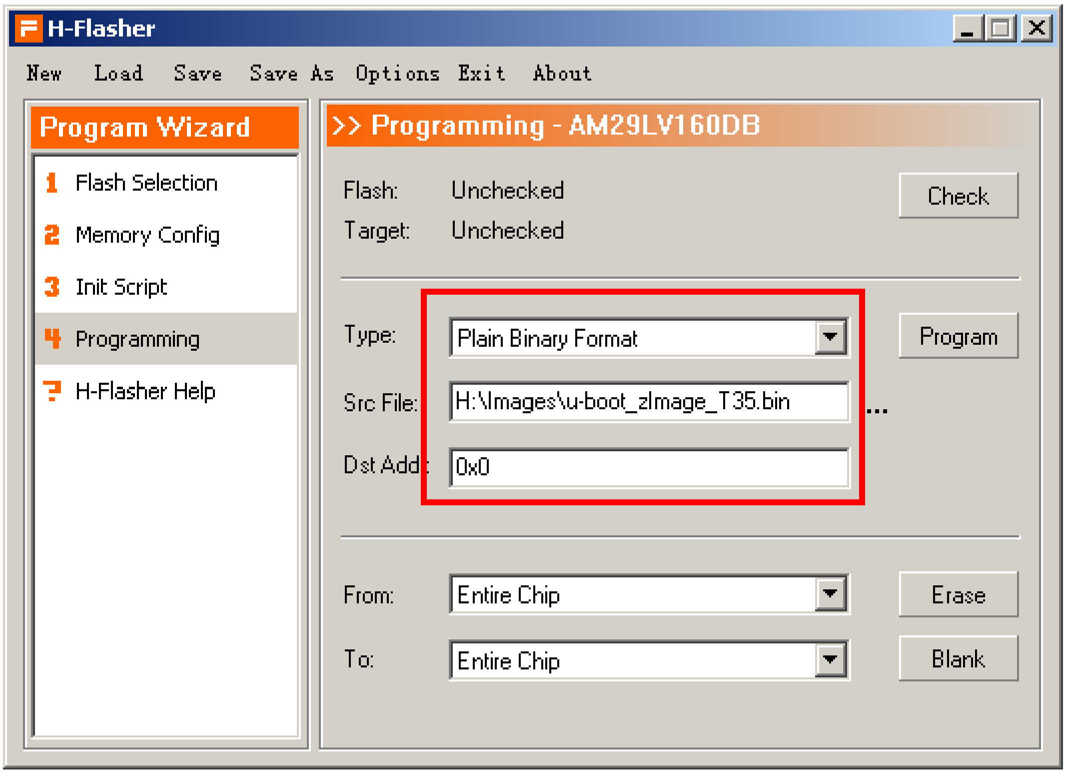
Step6, click “3 Init Script” leftward and click the adding button “ ” as the following diagram:



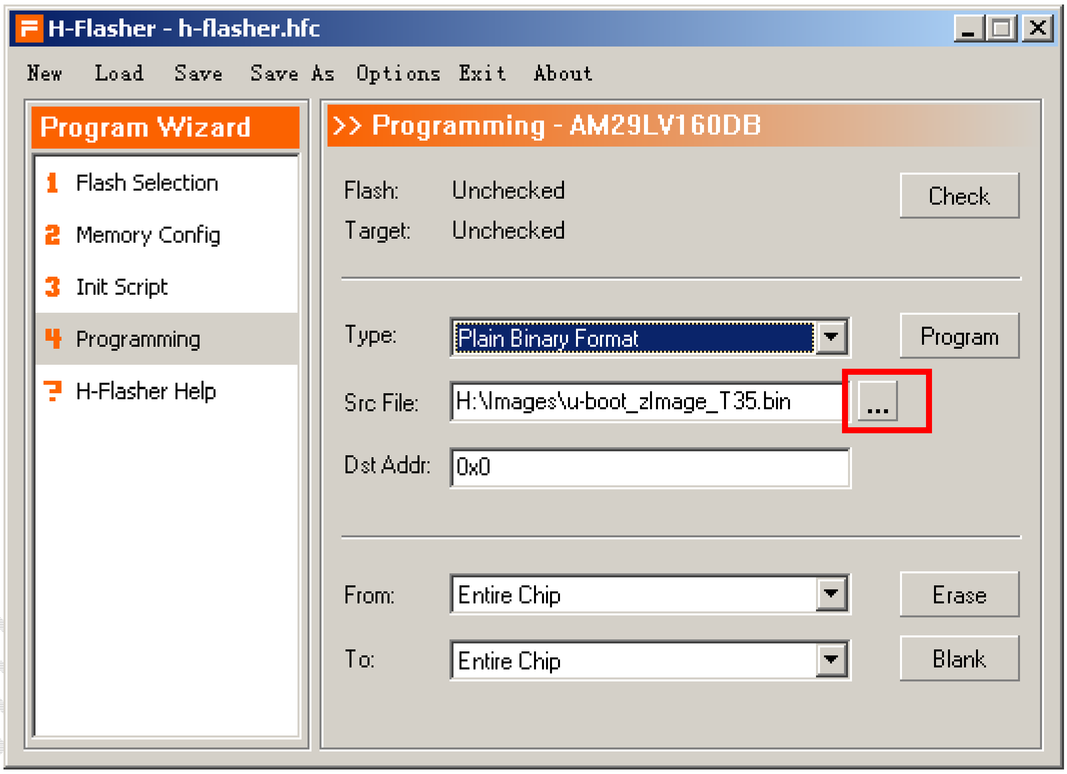
Step7, click “4 Programming”. The following interface appears:



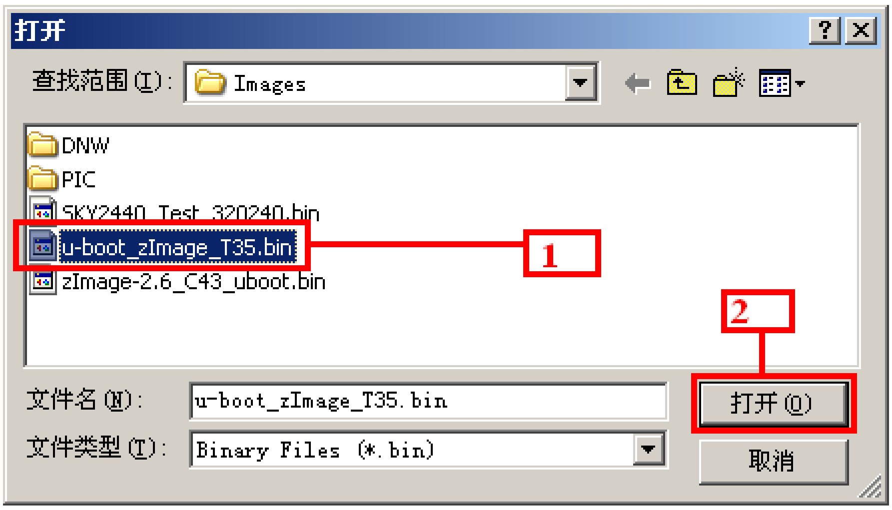
Make configuration as the following diagram:



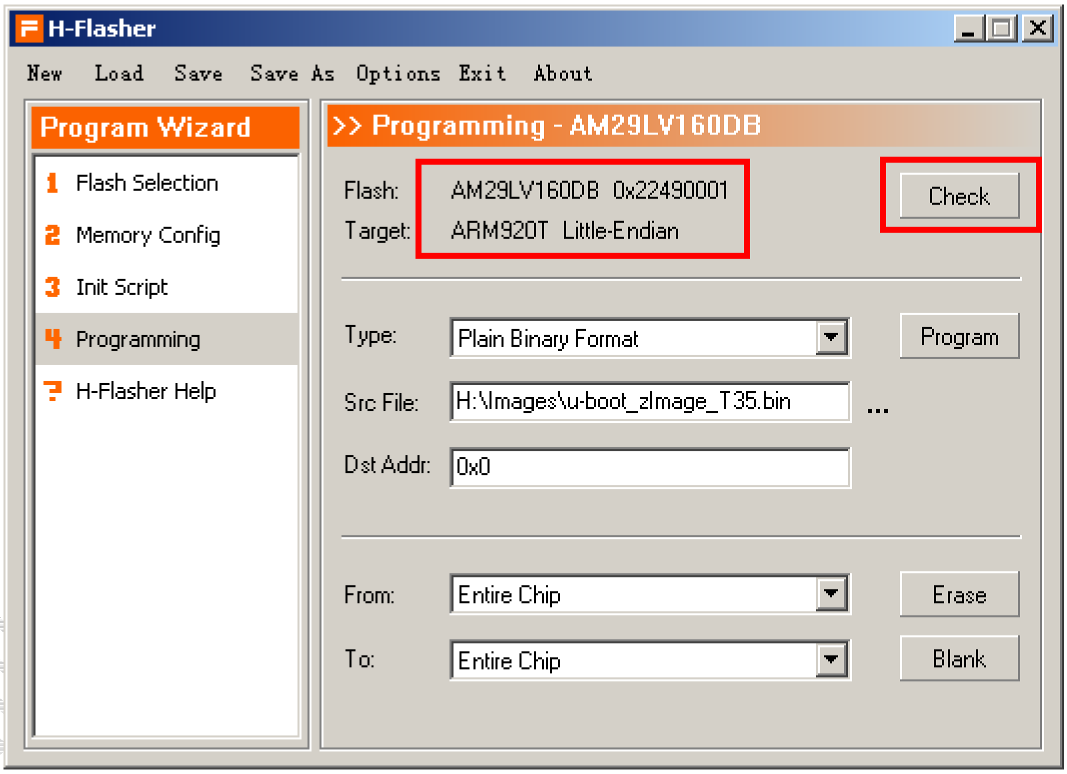
Click "...":



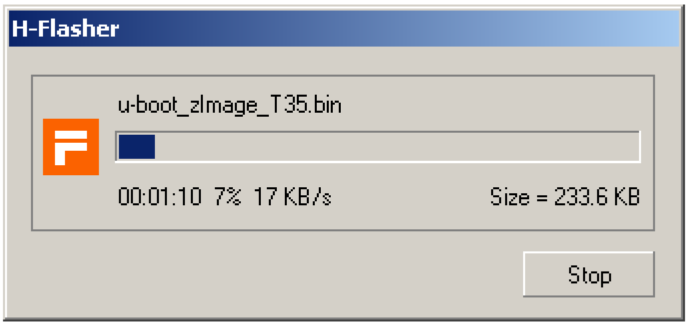
Locate u-boot:



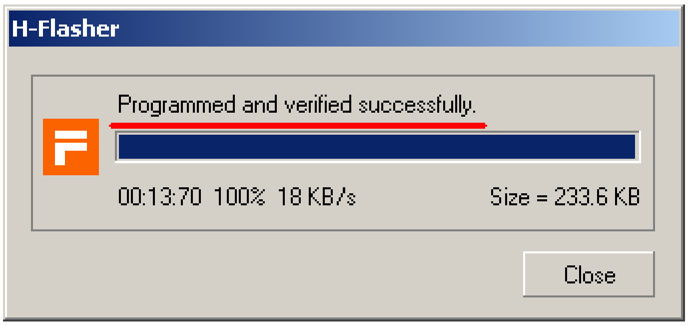
Click "Check" rightward. You can find the CPU ID and Nor Flash in the window:



Step8, burning u-boot. Click "Program" to begin burning:



Click "Close" to finish burning.



Step9, click "save" and there is no need to configure H-Flasher again next time.

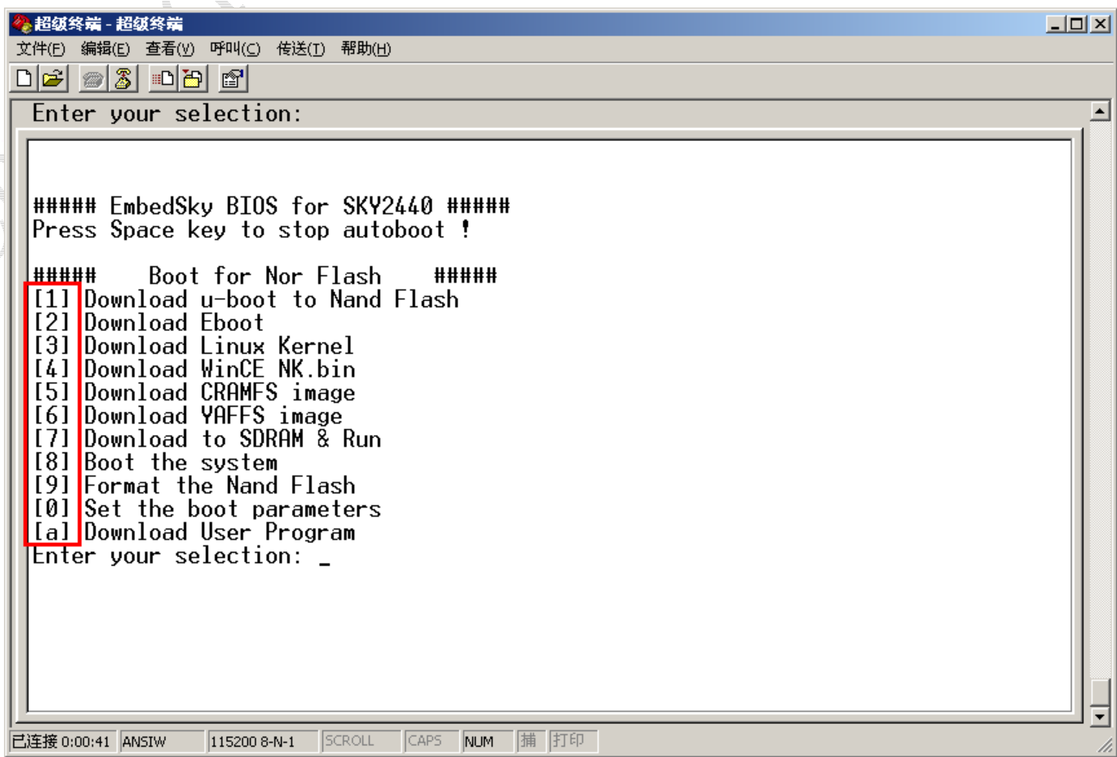
3. 4 u-boot application

u-boot in TQ2440 supports 2 start-up modes, starting from Nor Flash or Nand Flash, and also supports 2 file download modes, download Linux and WinCE by using USB or TFTP.

The following contents illustrate how to use one-key operation menu. One-key operation menu adopts USB download mode as default. If you have selected TFTP download mode, please exit one-key operation menu and enter into the default operation interface of u-boot.

3. 4. 1 Function introduction

The start-up interface is shown in the following diagram. The operation code is located inside the red frame:



3. 4. 2 burning u-boot

Please consult “2.3 节” and “2.4 节”

3. 4. 3 Configuration and estimation of u-boot start-up state

There are 3 states to select when TQ2440 starts up:

- State 1：Download mode. Update u-boot, Linux and WinCE in this mode;
- State 2：Linux start-up mode. Load and boot Linux in this mode;

- State 3：WinCE start-up mode. Load and boot WinCE in this mode.

Entering into state 1: Connect serial port of platform, press and hold space-key of PC keyboard, and then switch on platform power. After that, “Donwload System mode” appears on LCD;

Entering into state 2 and state 3: Switch on power directly. System starts boot Linux or WinCE based on the software installed in platform. The phase “Linux System Loading …” (state 2) or “WinCE System Loading …” (state 3) appears on LCD soon.

3. 4. 4 Introduction of u-boot utilization

There are 11 kinds of operations when TQ2440 u-boot starts up.

Caution: The following introduction about TQ2440 u-boot utilization is arranged in accord with the menu sequence. Please consult “chapter 3.7” when burning Linux or WinCE.

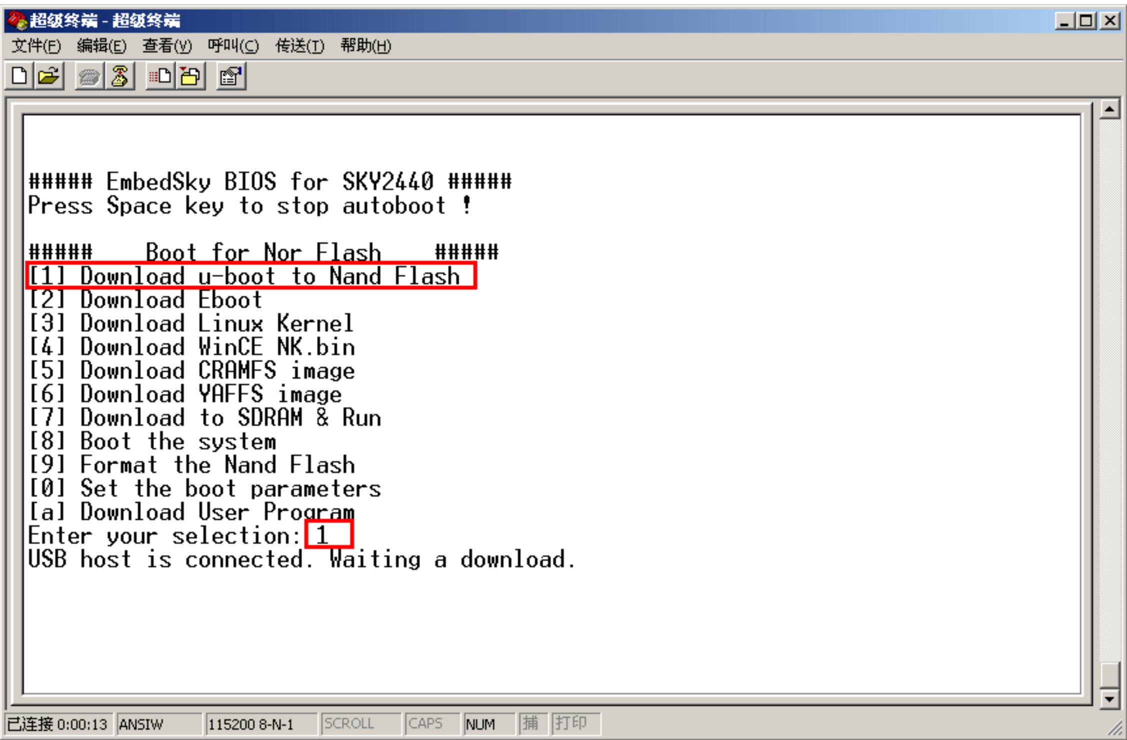
The way entering into u-boot console: Start Hyper Terminal, press and hold space-key of PC keyboard, and switch on platform power.

Caution: eboot and Linux kernel partly overlap. Please burn eboot first if you try to change Linux into WinCE.

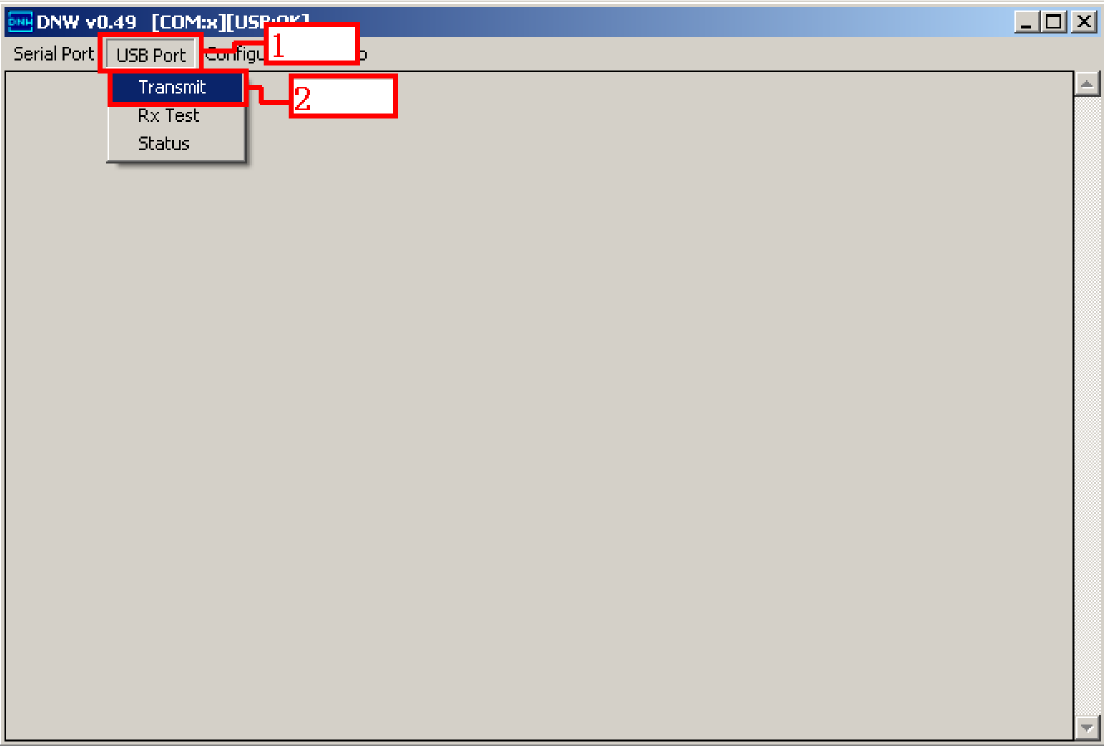
The files referred in the following contents are included under the directory of “Images\Linux” or “Images\WinCE” in CD-ROM

Step 1, enter “1”: Download u-boot to Nand Flash, as shown in the following diagram:

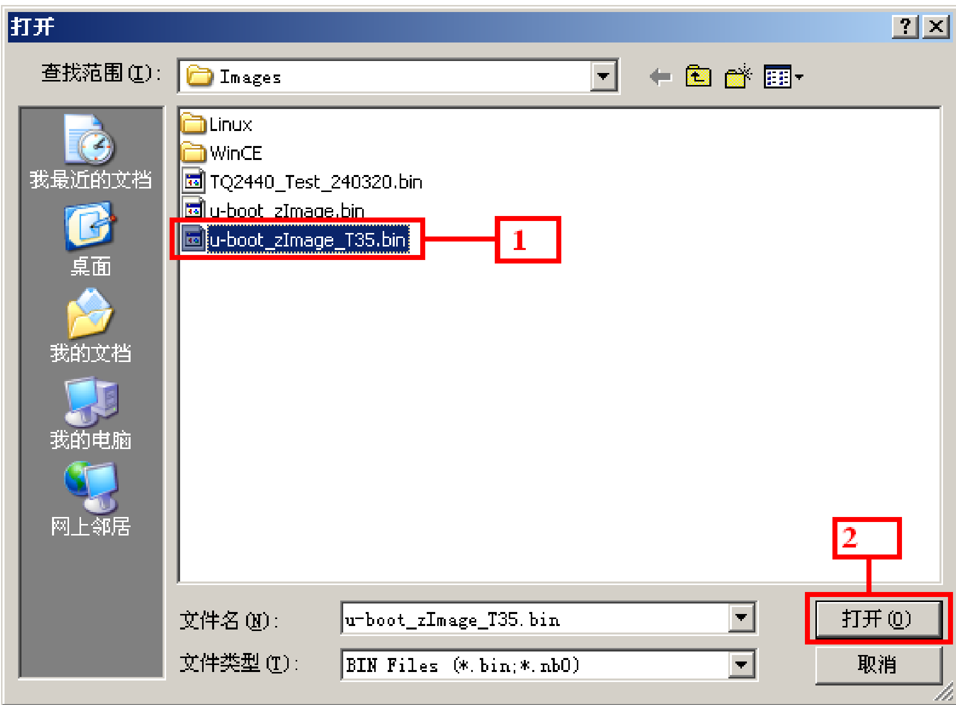
(caution: This operation is corresponding to reading and writing Nand Flash, if u-boot works well, it is not suggested to re-burn u-boot.)



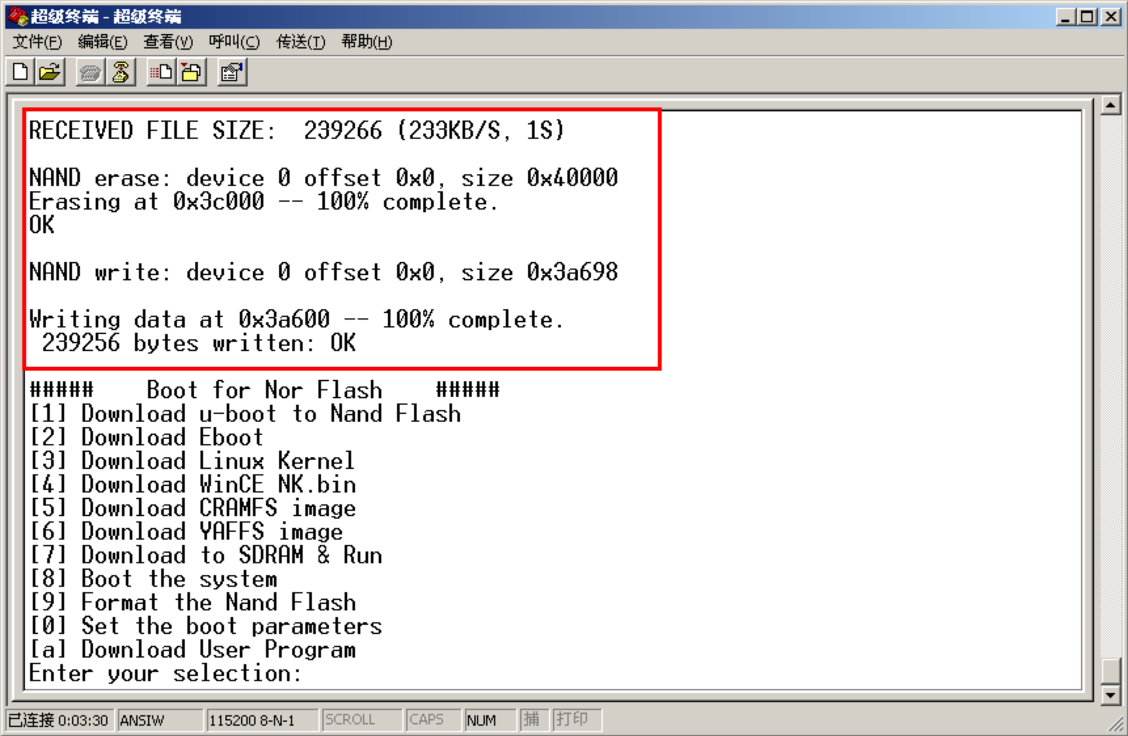
Open the software DNW when the upper sentence “USB hosting is connected, Waiting a download.” appears. Click “Transmit” in “USB Port” menu:



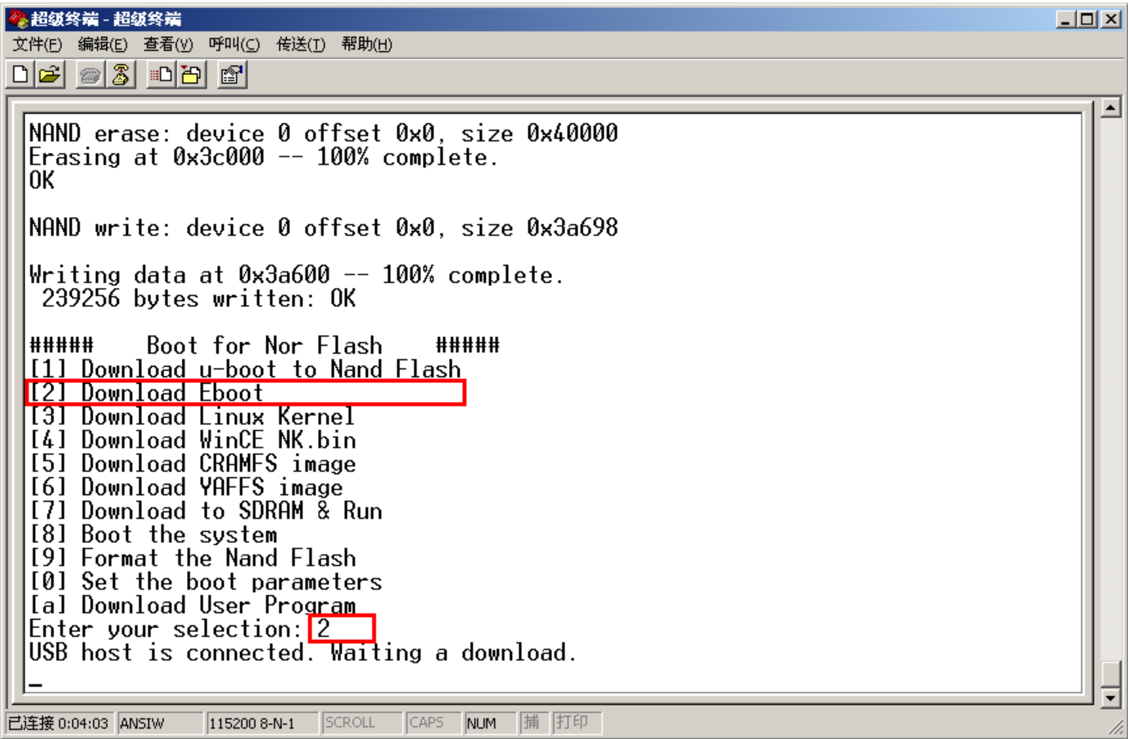
Locate u-boot in the following interface. Find “u-boot_zImage_xxxx.bin” under the directory of “Image\Linux” or “Image\WinCE” in CD-ROM (xxxx represents LCD type, T35 is Toshiba 3.5 inch LCD, S35 is Sumsung 3.5 inch LCD and S70 is Sumsung 7 inch LCD). Click “打开” to continue:



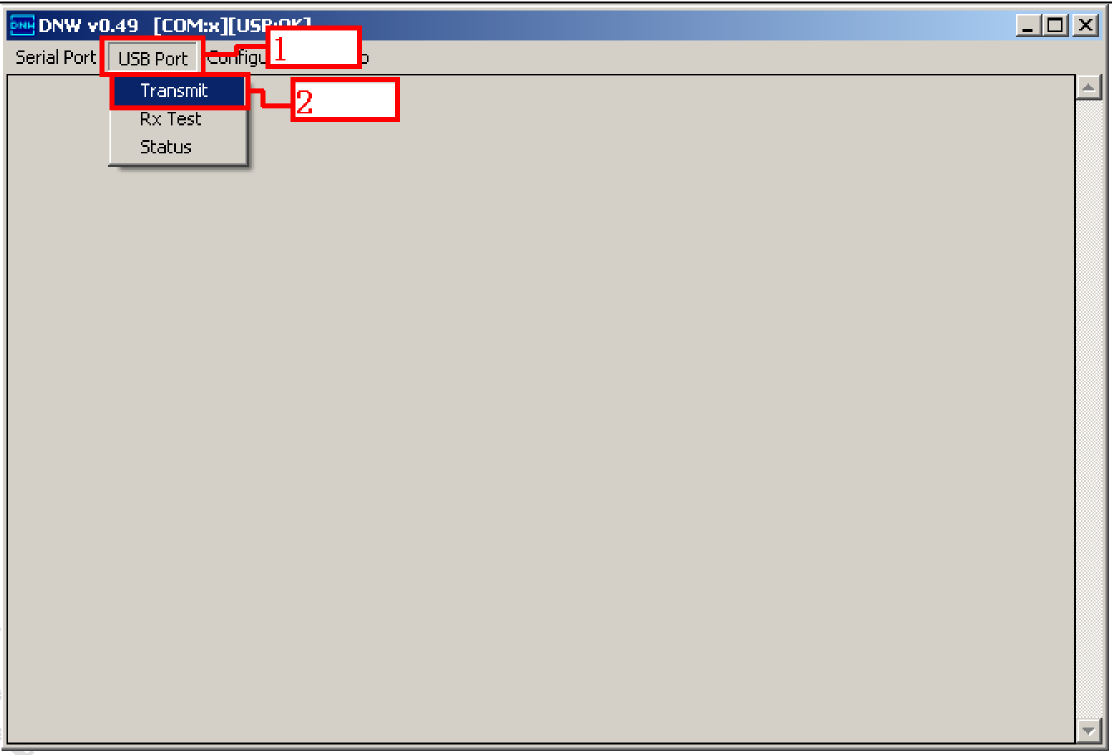
u-boot is downloaded and saved automatically:



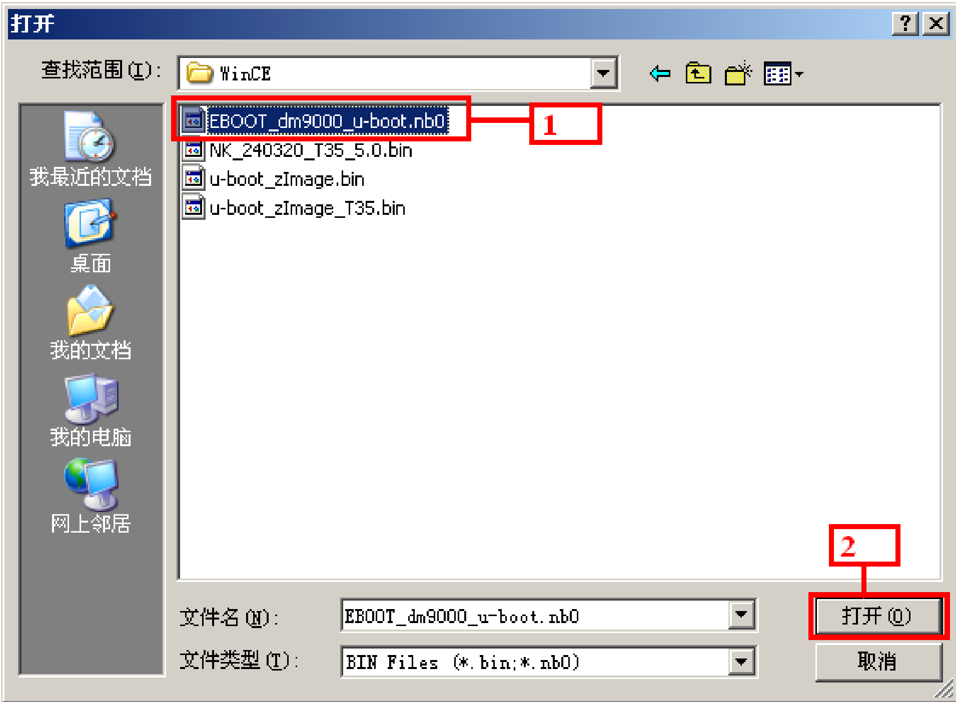
Step 2, enter “2”: Download eboot, as shown in the following diagram:



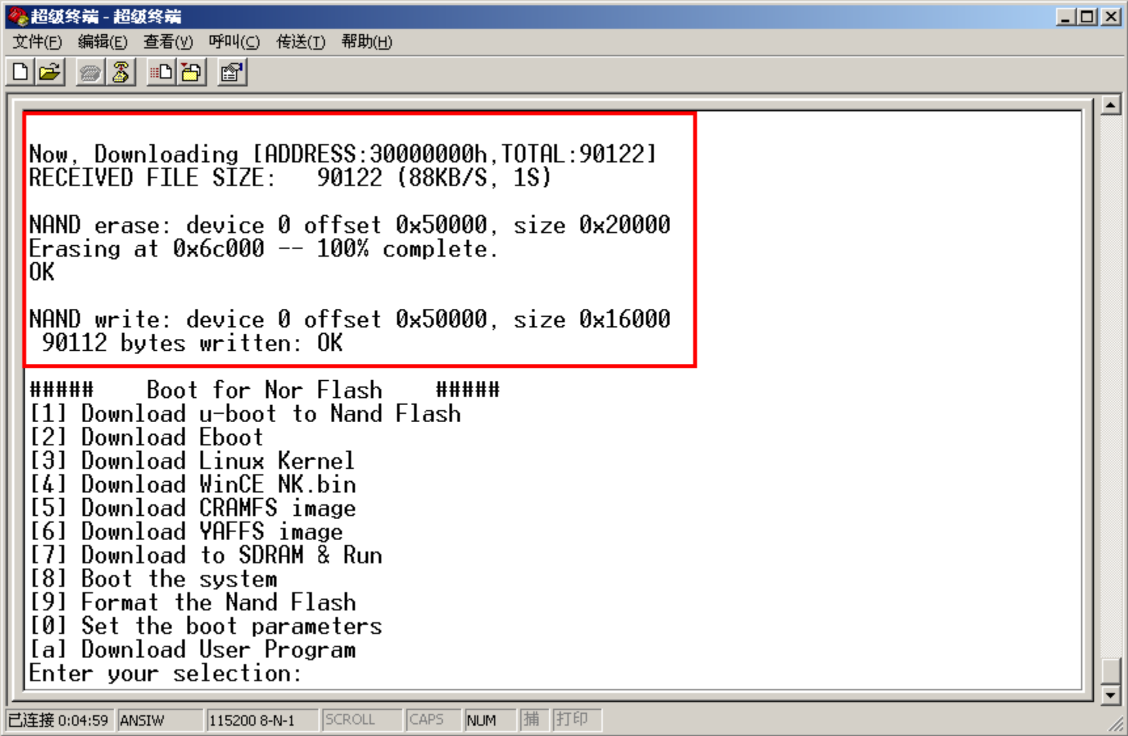
Open the software DNW when the upper sentence “USB hosting is connected，Waiting a download.” appears. Click “Transmit” in “USB Port” menu:



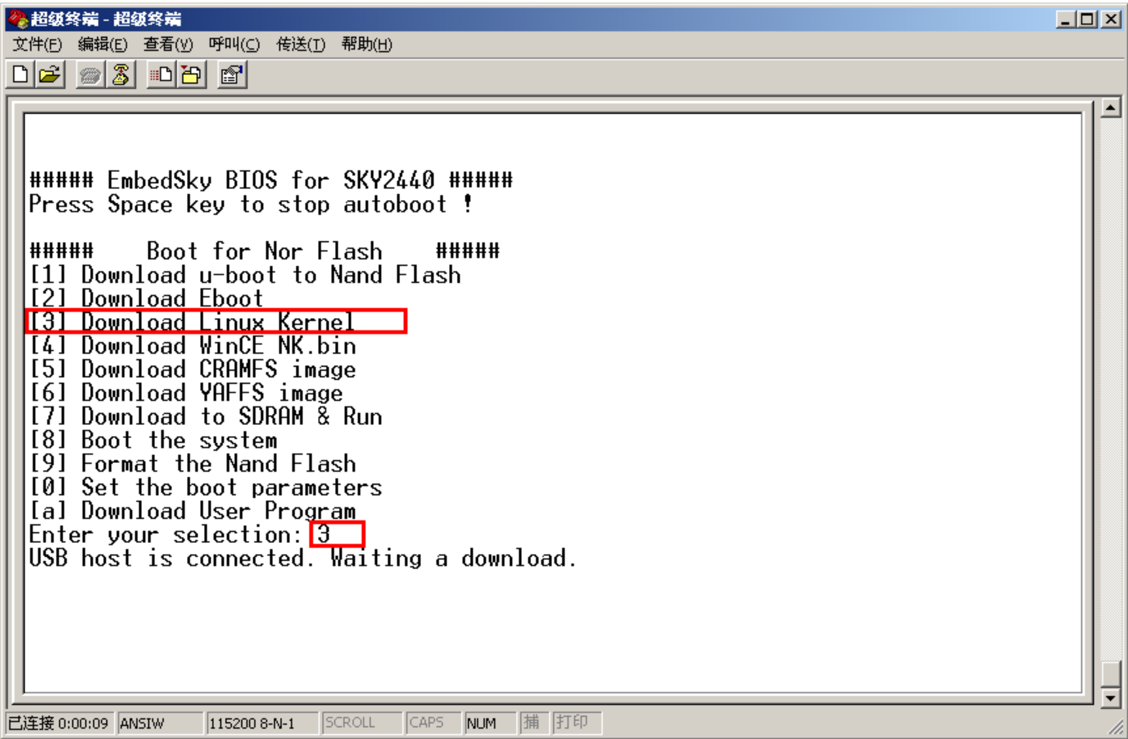
Locate "EBOOT_dm9000_u-boot.nb0" under the directory of "Images\WinCE" in CD-ROM and click "打开" to continue:



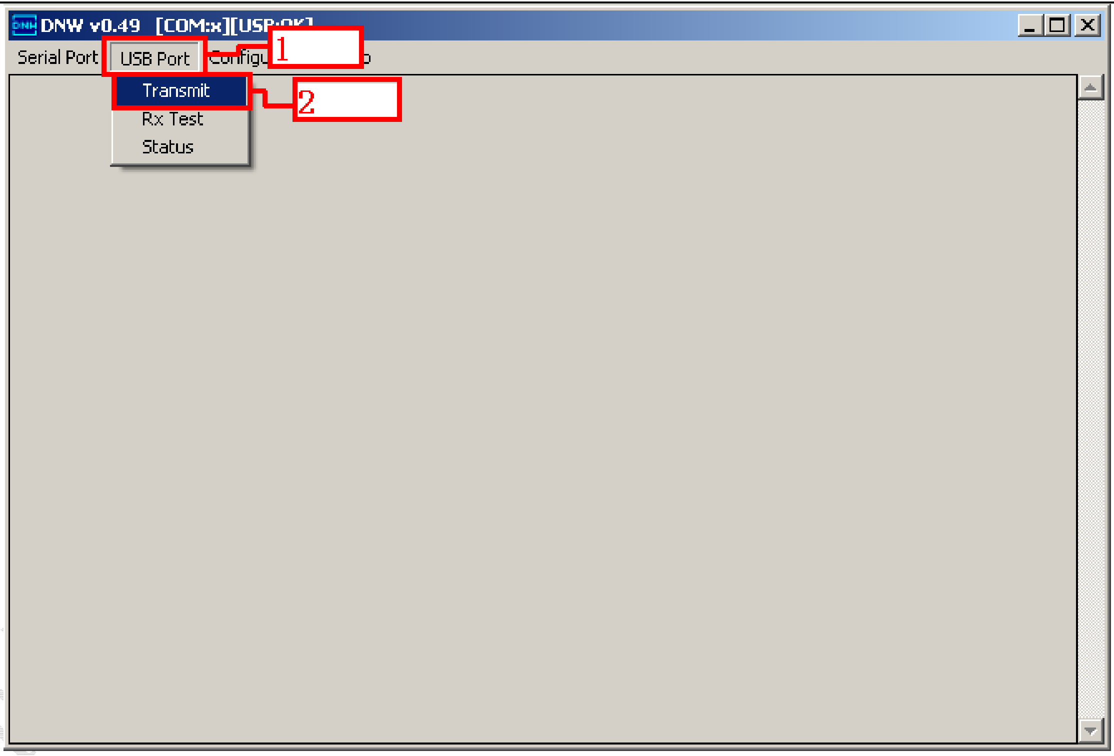
eboot is downloaded and saved automatically:



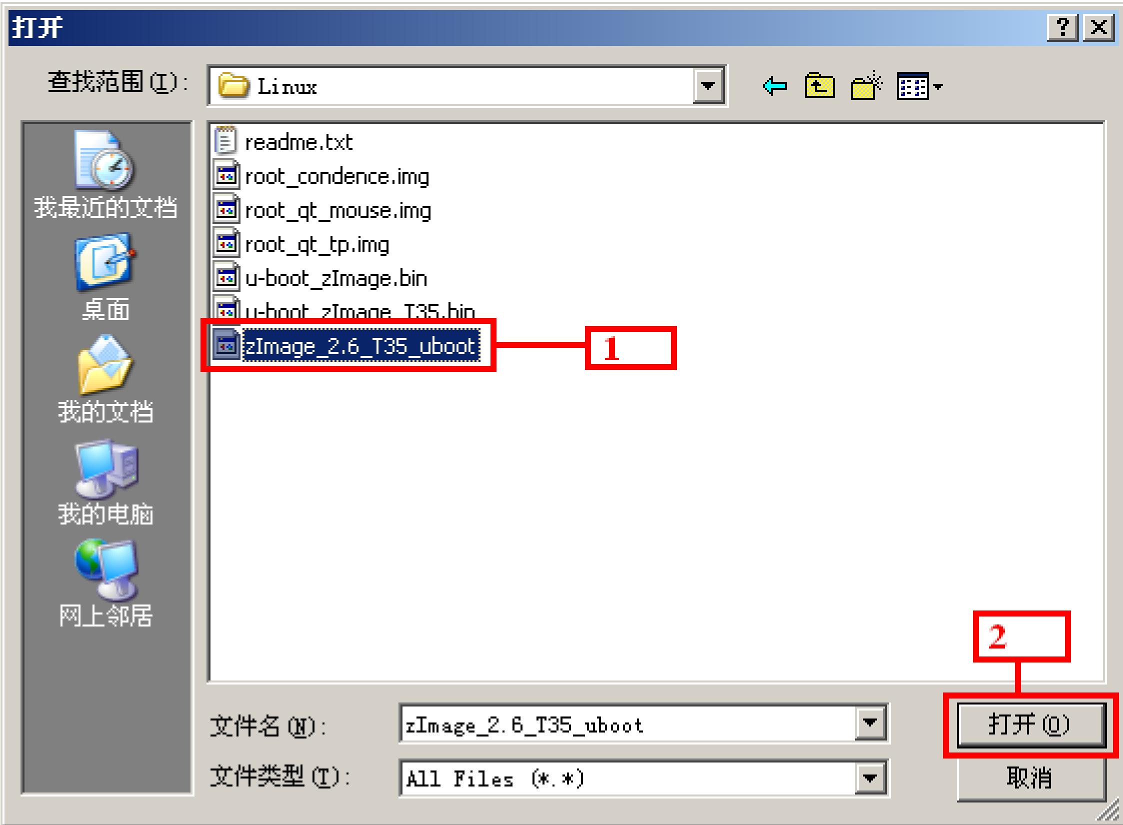
Step 3, enter “3”: Download Linux kernel, as shown in the following diagram:



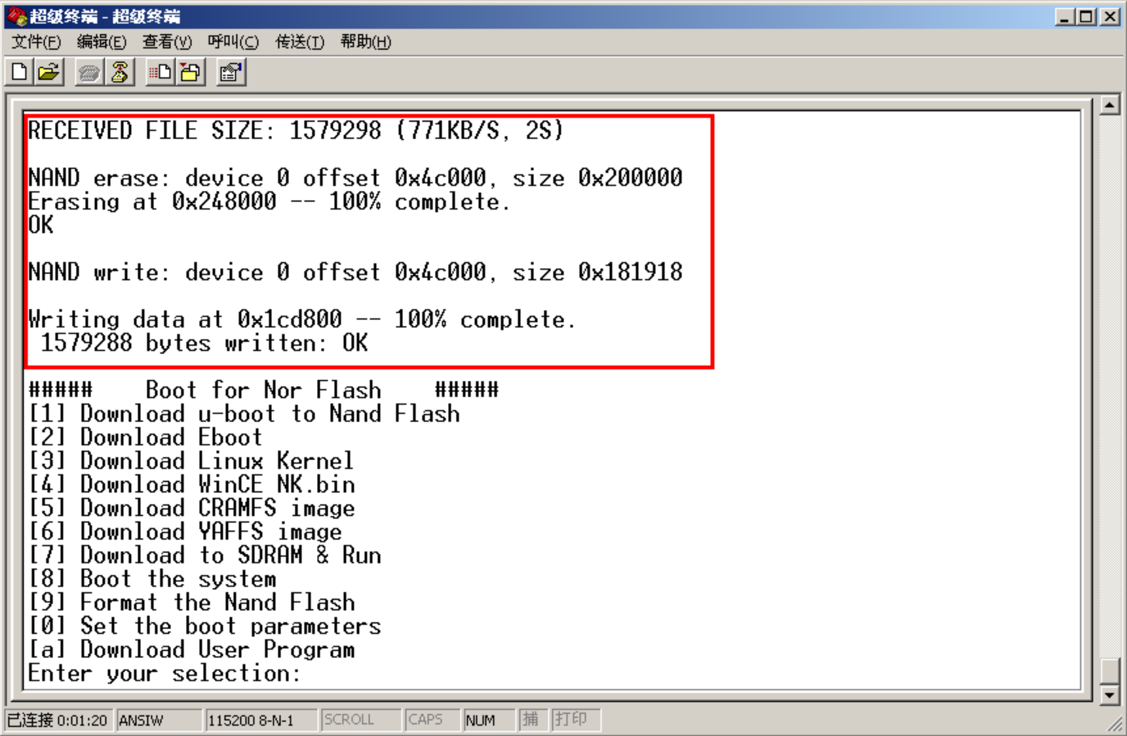
Open the software DNW when the upper sentence “USB hosting is connected，Waiting a download.” appears. Click “Transmit” in “USB Port” menu:



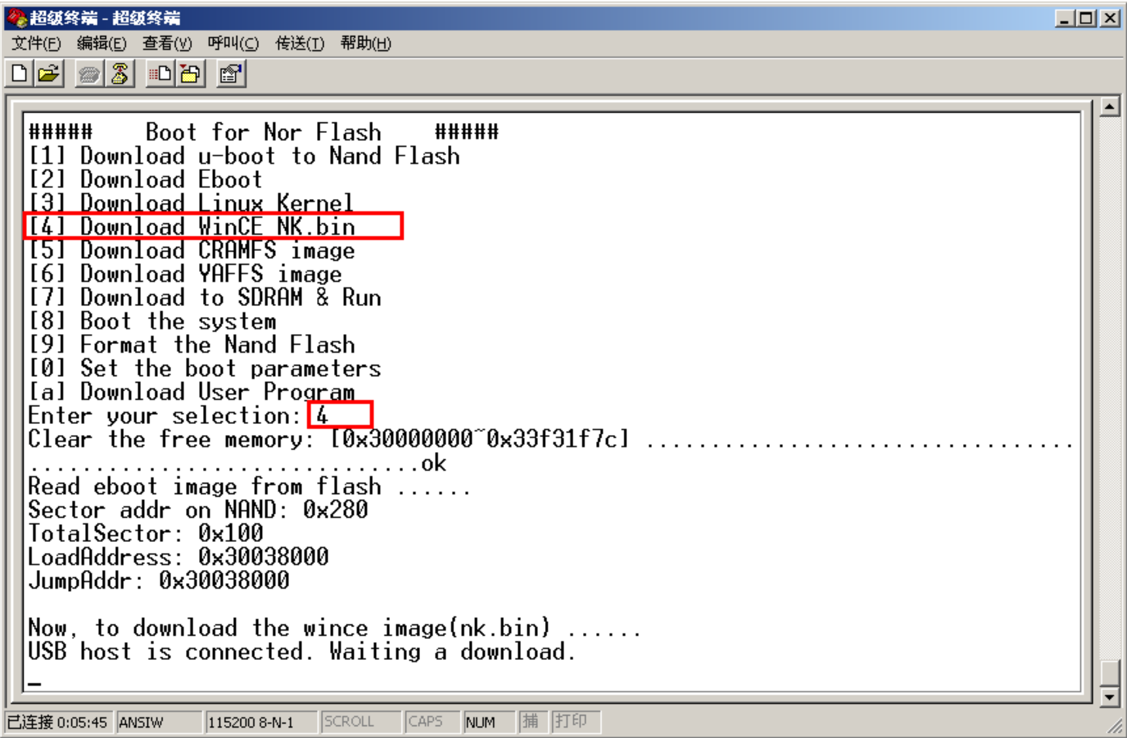
Locate "zImage_2.6_xxxx_uboot" under the directory of "Image\Linux" in CD-ROM(xxxx represents LCD type, T35 is Toshiba 3.5 inch LCD, S35 is Sumsung 3.5 inch LCD, W35 is Donghua 3.5 inch LCD and S70 is Sumsung 7 inch LCD). Click "打开" to continue:



Linux kernel image is downloaded and saved automatically:

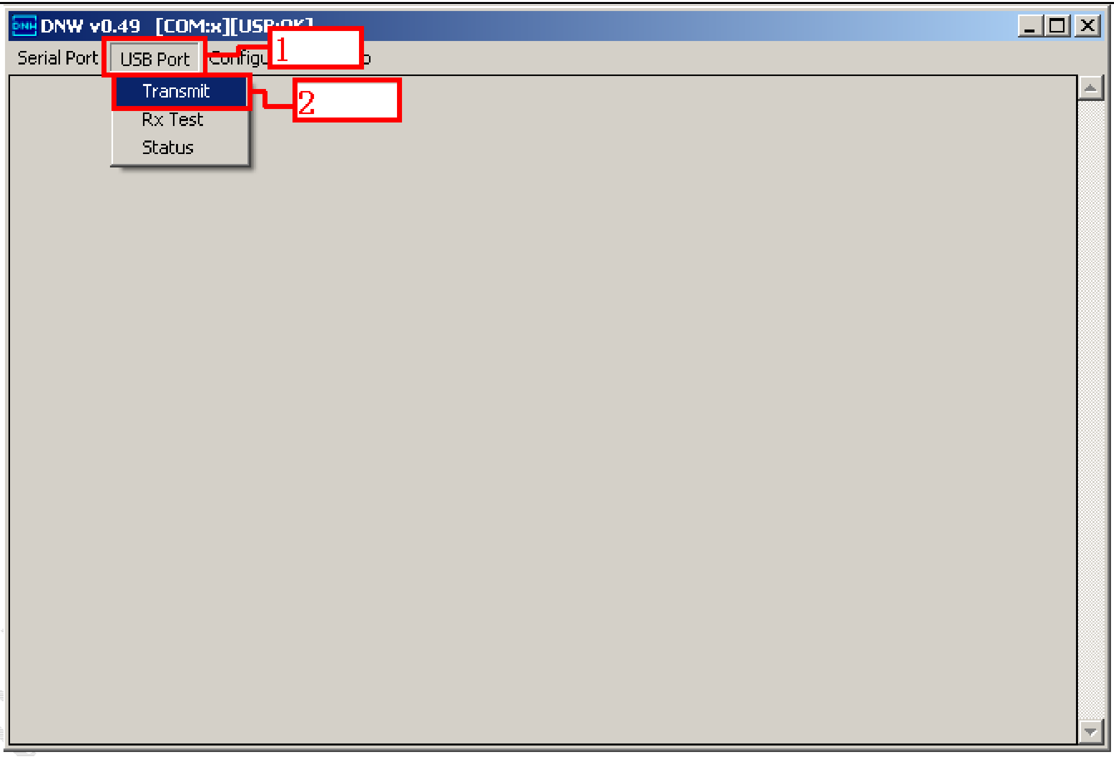


Step 4, enter “4”: Download WinCE NK.bin, as shown in the following diagram:

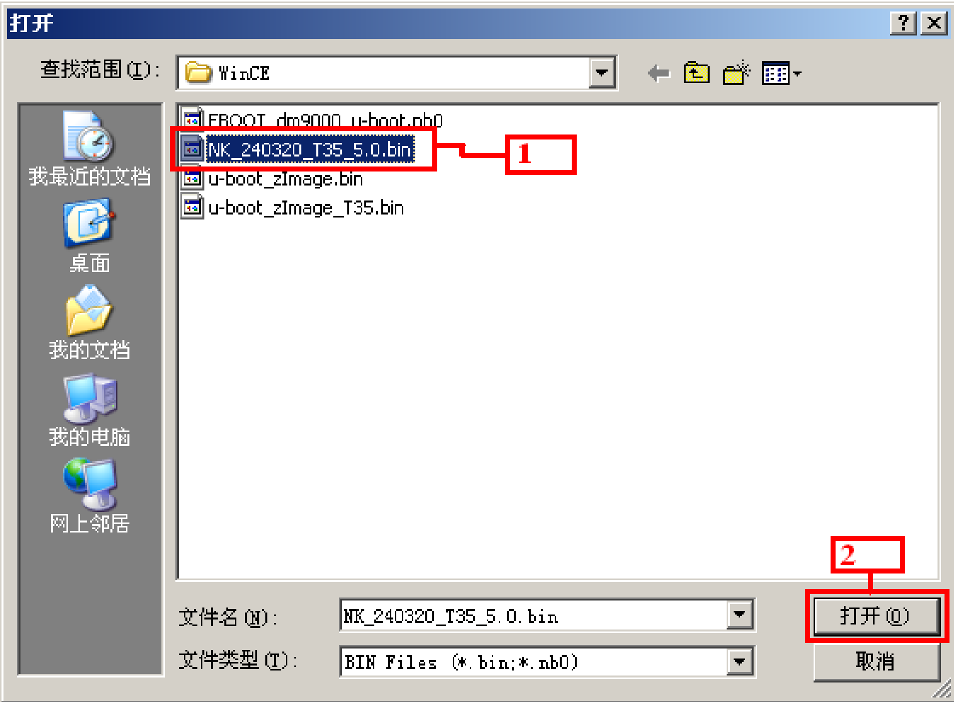


Open the software DNW when the upper window “USB hosting is connected，Waiting a download.” appears, and click “Transmit” in the menu “USB Port”

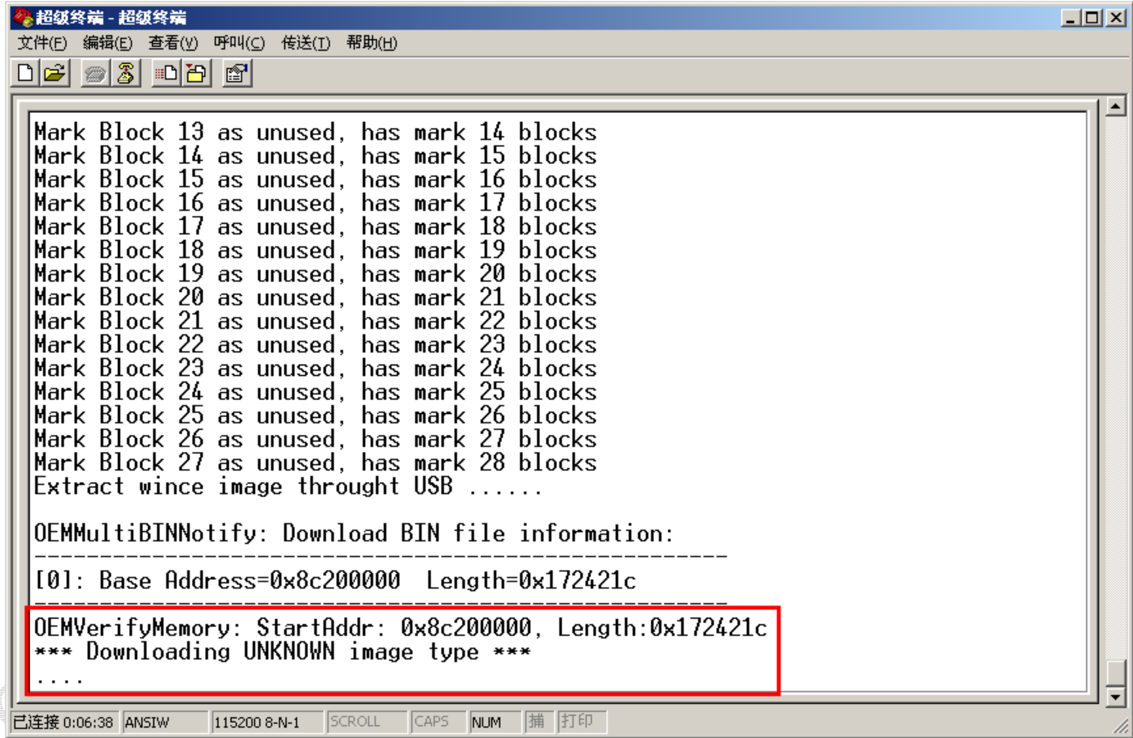
CuteDigi



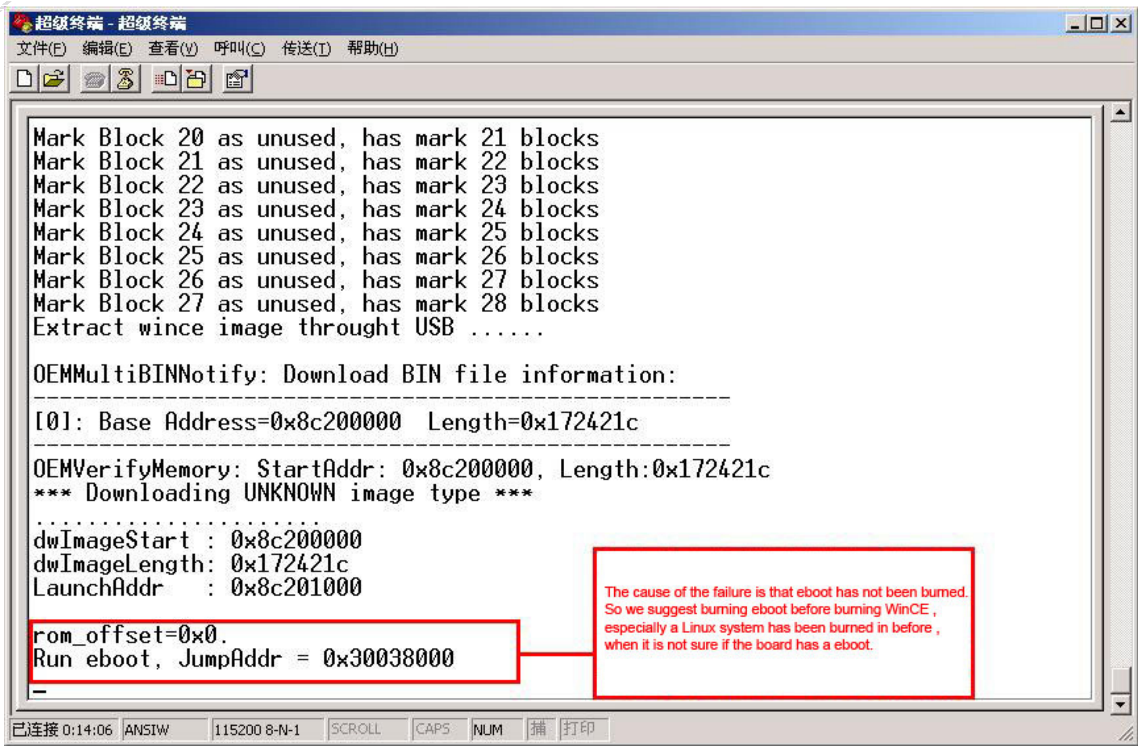
Locate “NK_xxxx_5.0.bin” under the directory of “Image\WinCE” in CD-ROM (xxxx represents LCD type, 240320_T35 is Toshiba 3.5 inch LCD, 320240_S35 is Sumsung 3.5 inch LCD, 320240_W35 is Donghua 3.5 inch LCD and 800480_S70 is Sumsung 7 inch LCD). Click “打开” to continue:



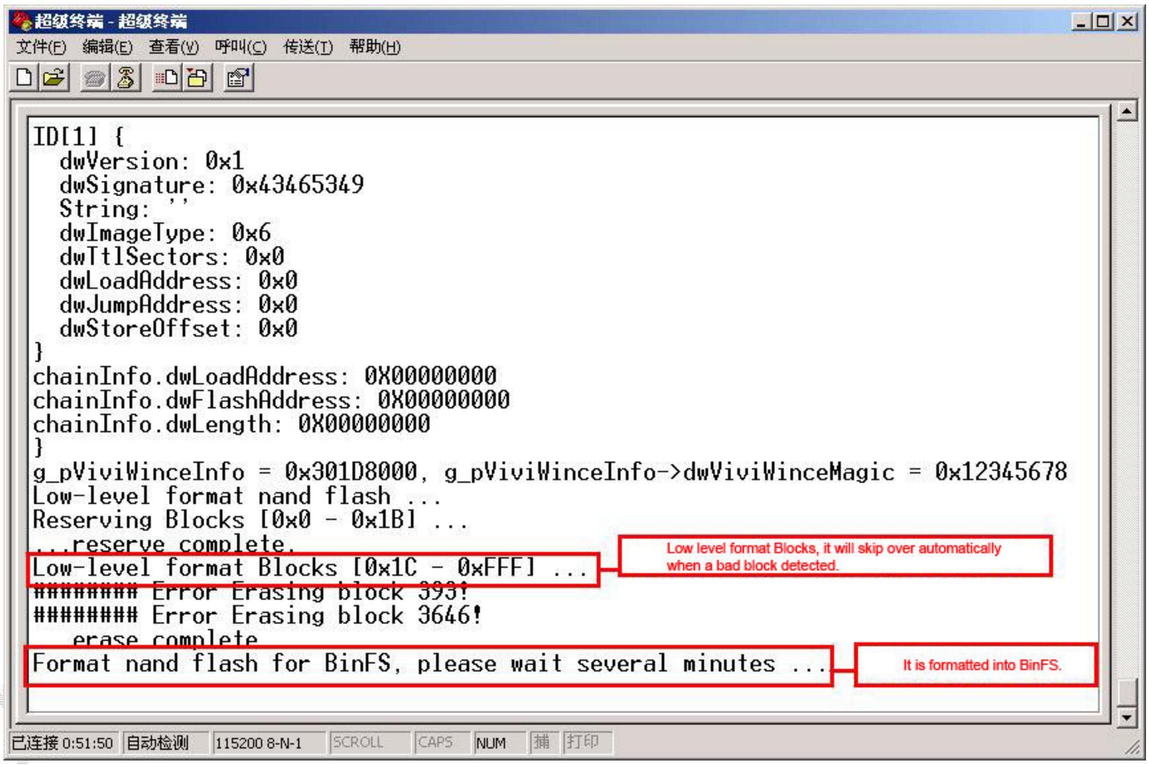
NK.bin image is being transmitted to memory in the following diagram:



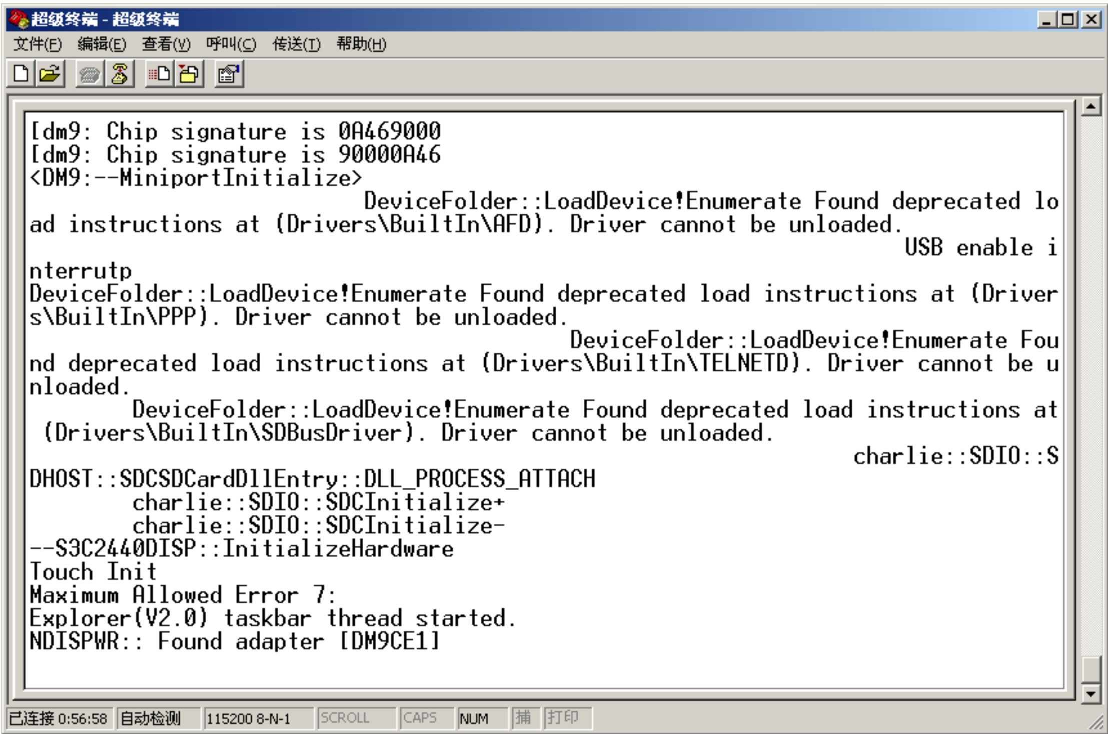
The following diagram displays a case that eboot has not been burned before the upper operations.:



After “NK.bin” has been successfully transmitted, Low-level format begins. Finally it is formatted into BinFS file (Bad blocks might be detected during formatting. Bad blocks is inevitable when Nand Flash is been produced, but our softwares are robust enough to aviod the side-effect brought by bad blocks):

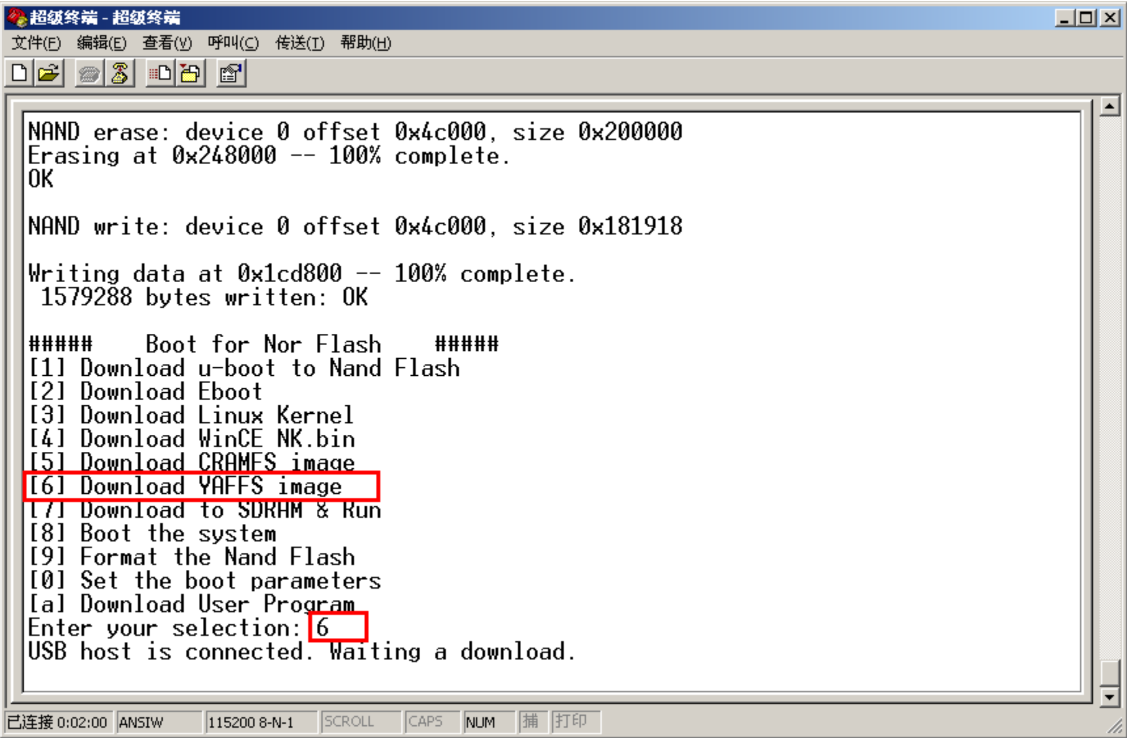


Burning WinCE is complete about 5 minutes later. Then WinCE starts up automatically:

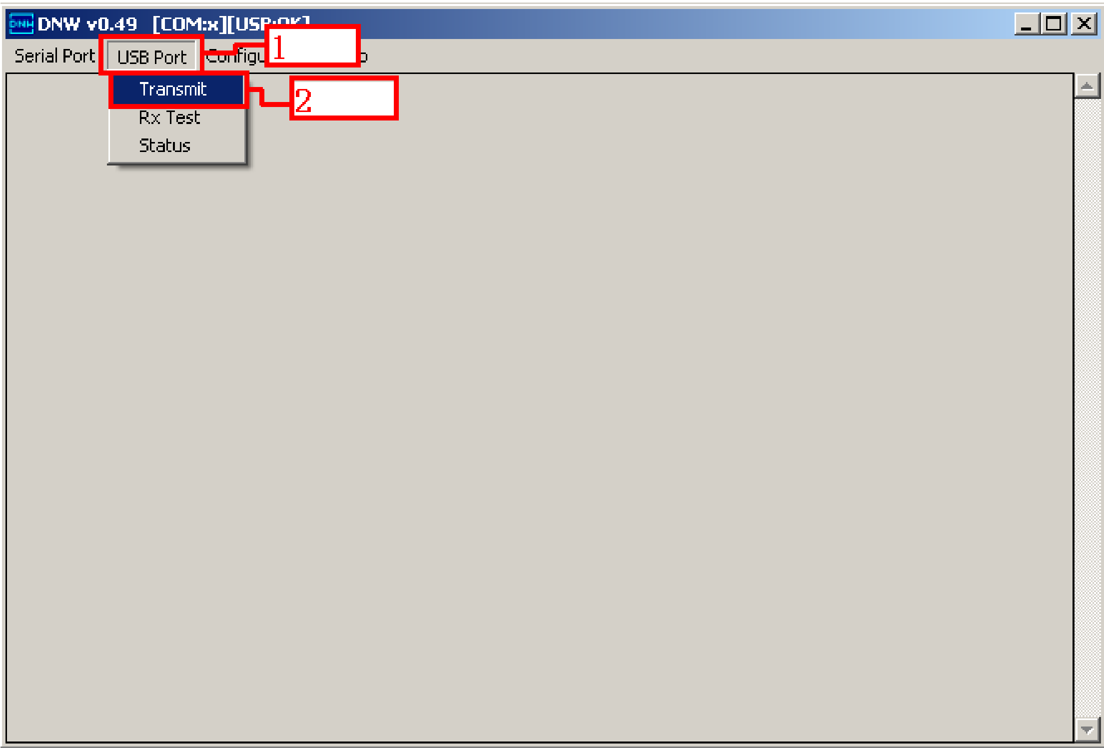


Step 5, enter “5”: Download Cramfs image (Cramfs is read-only which is not as convenient as Yaffs. So Cramfs is not recommended. The download process is the same as Yaffs)

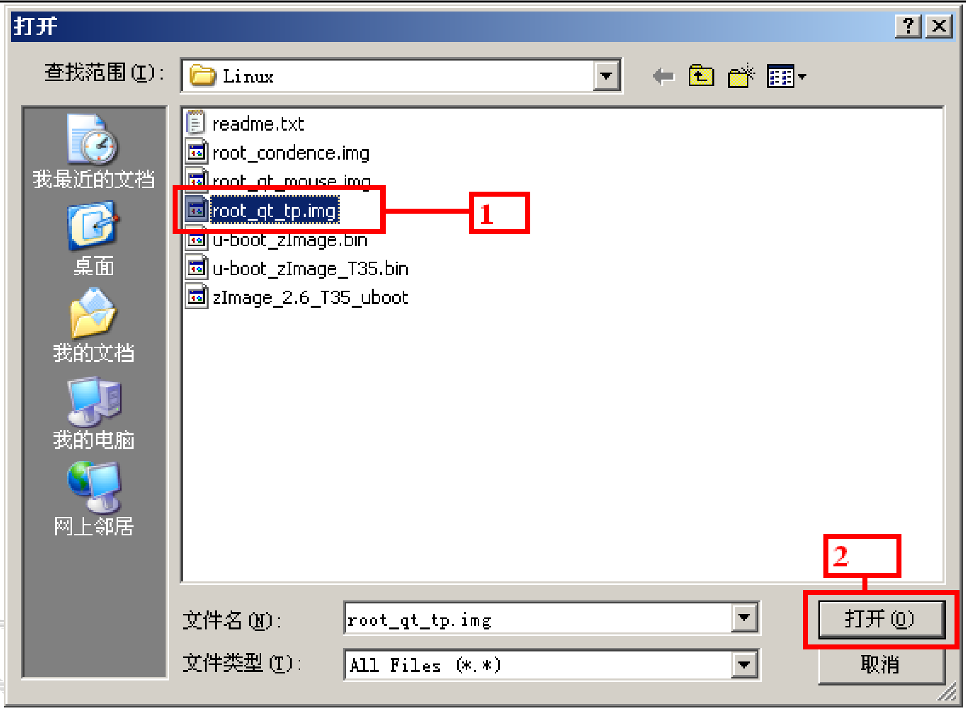
Step 6, enter “6”: Download Yaffs image, as shown in the following diagram:



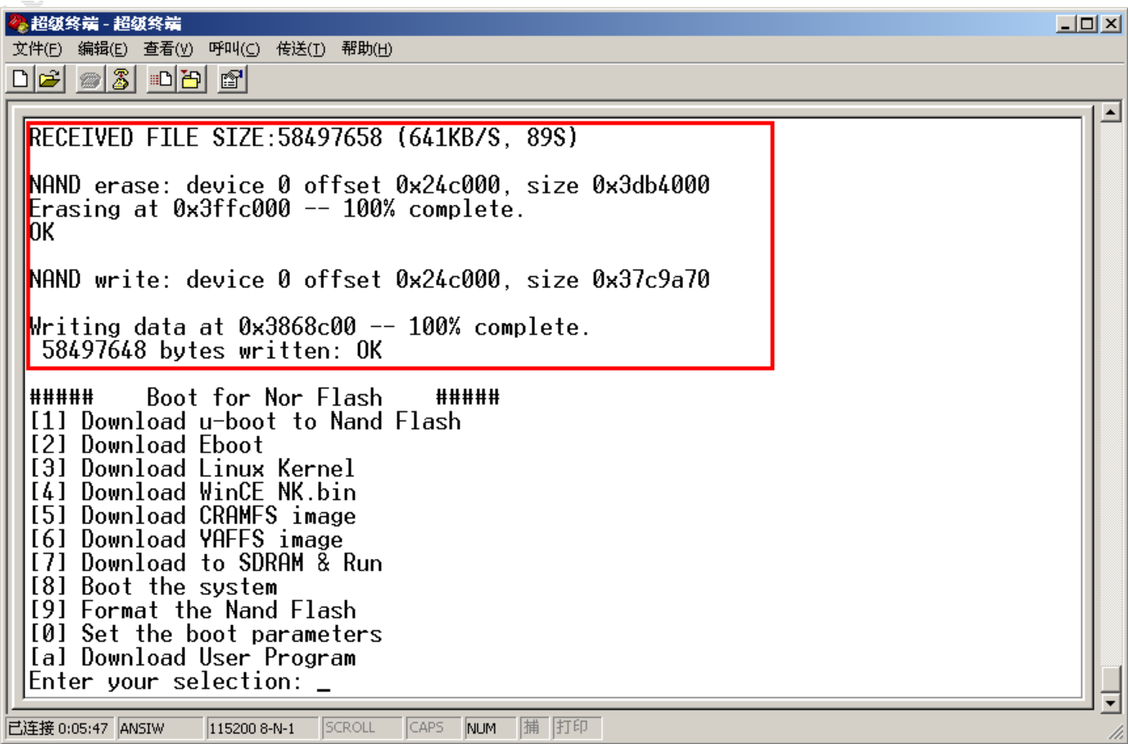
Open the software DNW when the upper sentence “USB hosting is connected，Waiting a download.” appears. Click “Transmit” in “USB Port” menu:



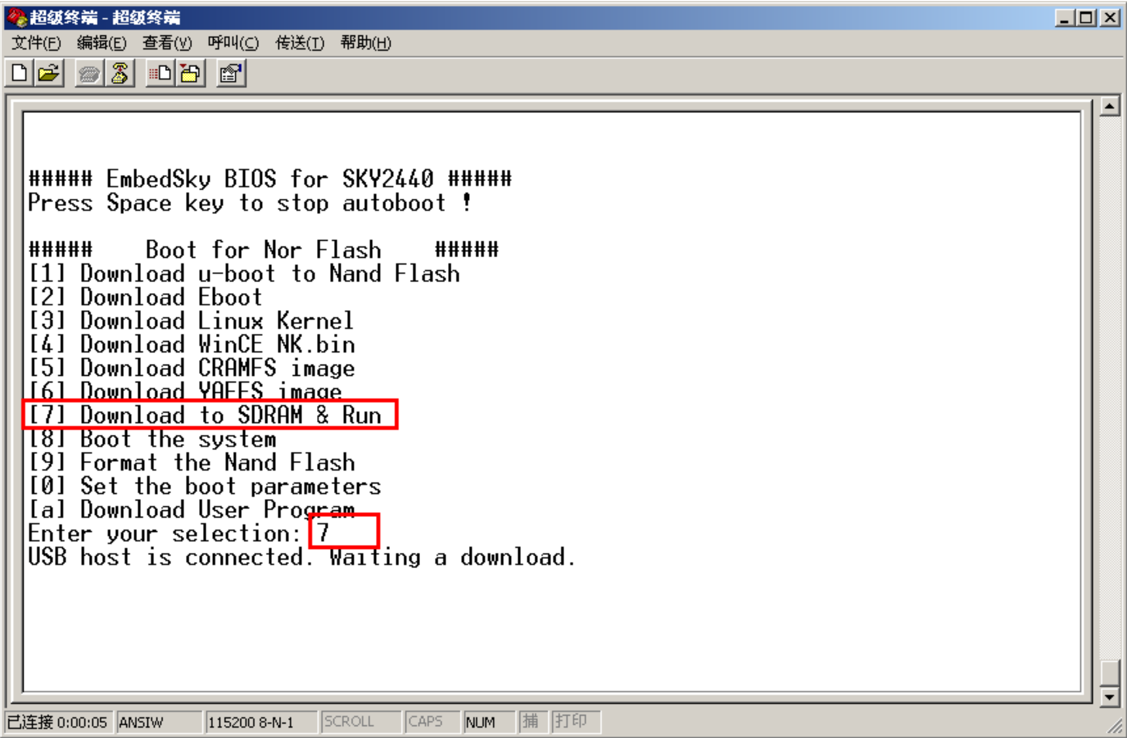
Locate “root_qt_tp.img” under the directory of “Image\Linux” in CD-ROM (root_condence.img is simplified file system; root_qt_mouse.img supports mouse and contains Qt file system; root_qt_tp.img supports touch-screen and contains Qt file system). Click “打开” to continue:



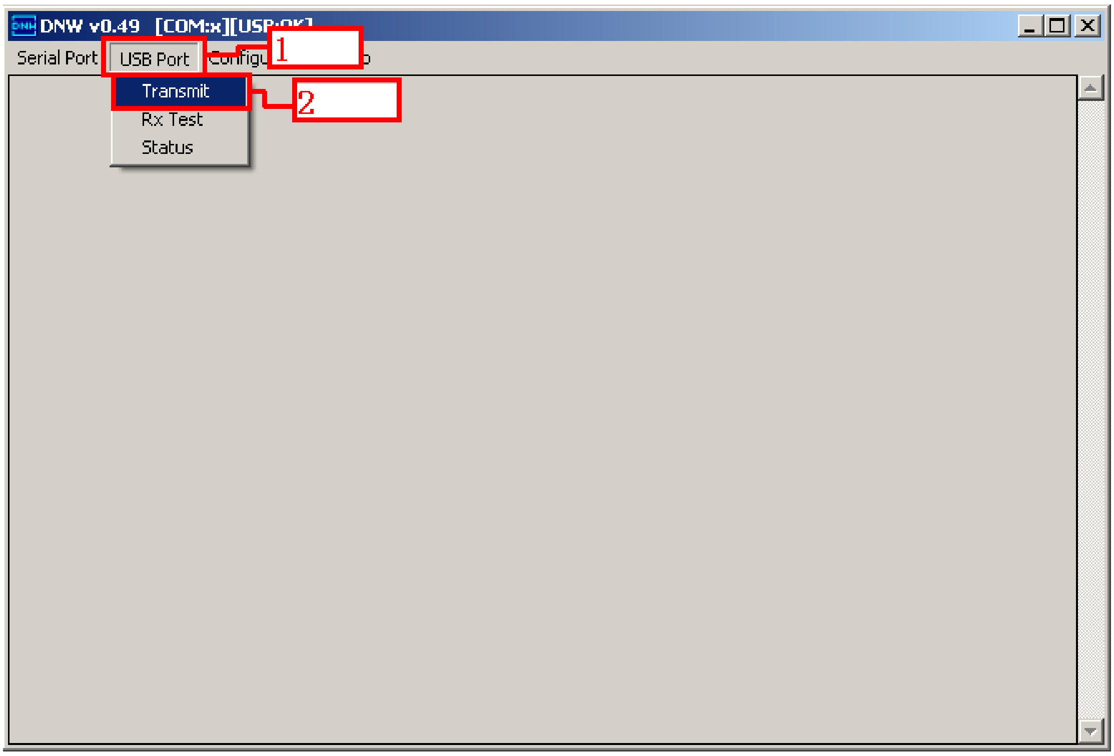
Yaffs file system is transferred and saved automatically:



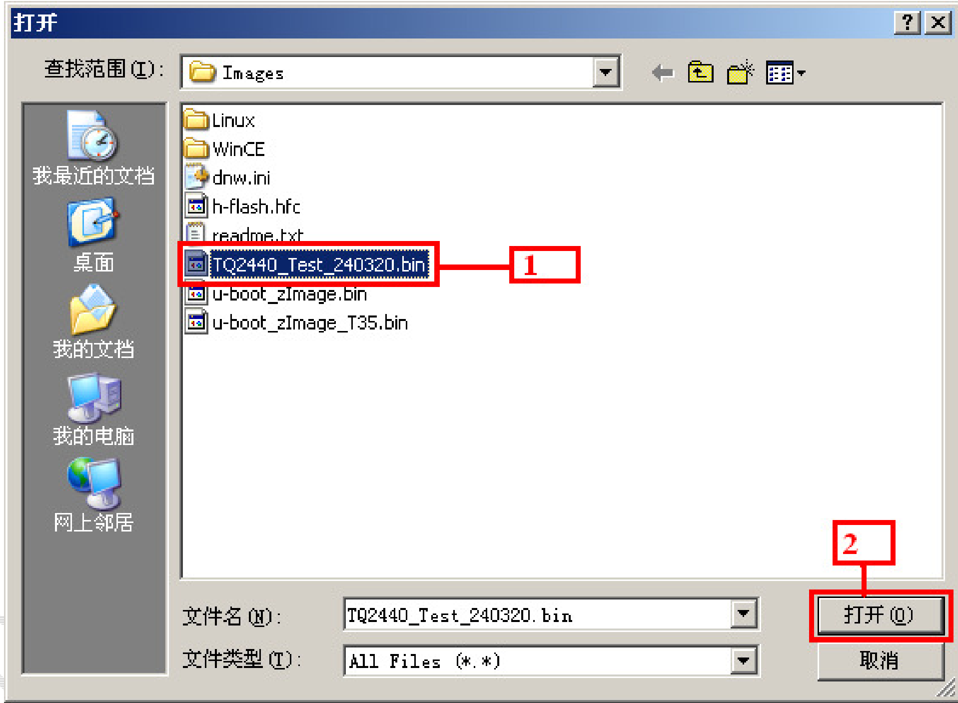
Step 7, enter “7”: Download to SDRAM & Run, as shown in the following diagram:



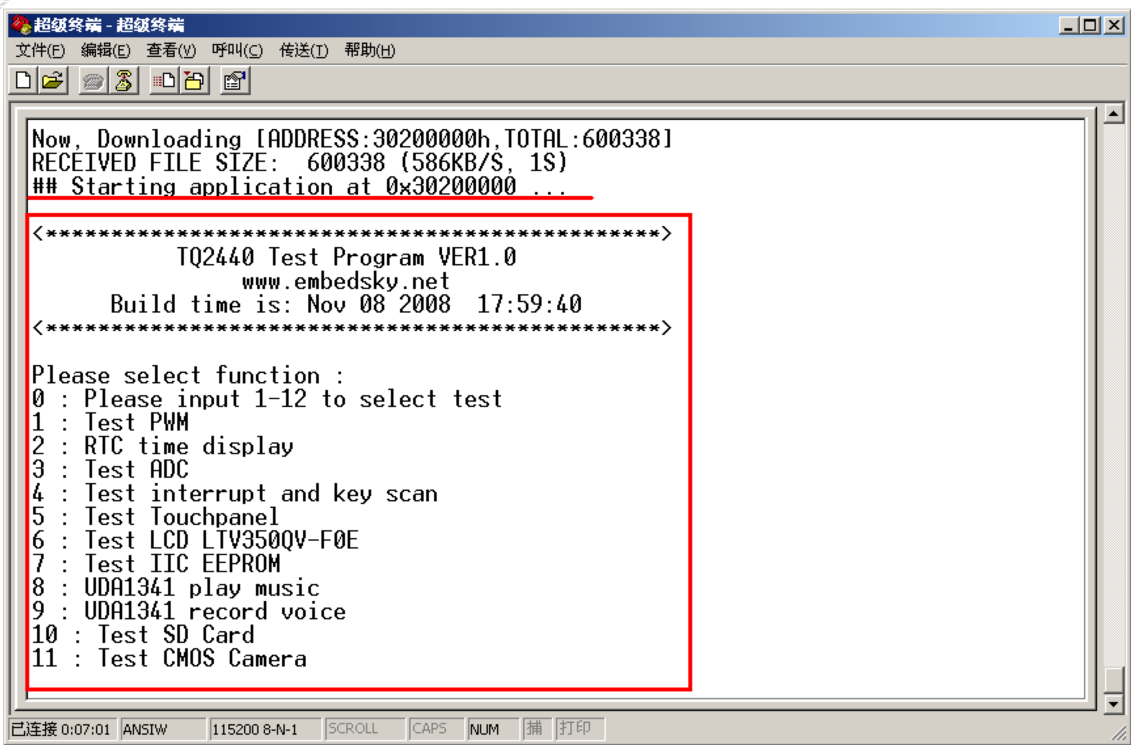
Open the software DNW when the upper sentence “USB hosting is connected，Waiting a download.” appears (you need to wait about 5 minutes for the sentence). Click “Transmit” in “USB Port” menu:



Locate “TQ2440_Test_xxxx.bin” under the directory “Images” in CD-ROM (xxxx represents LCD resolution). Click “打开” to continue:

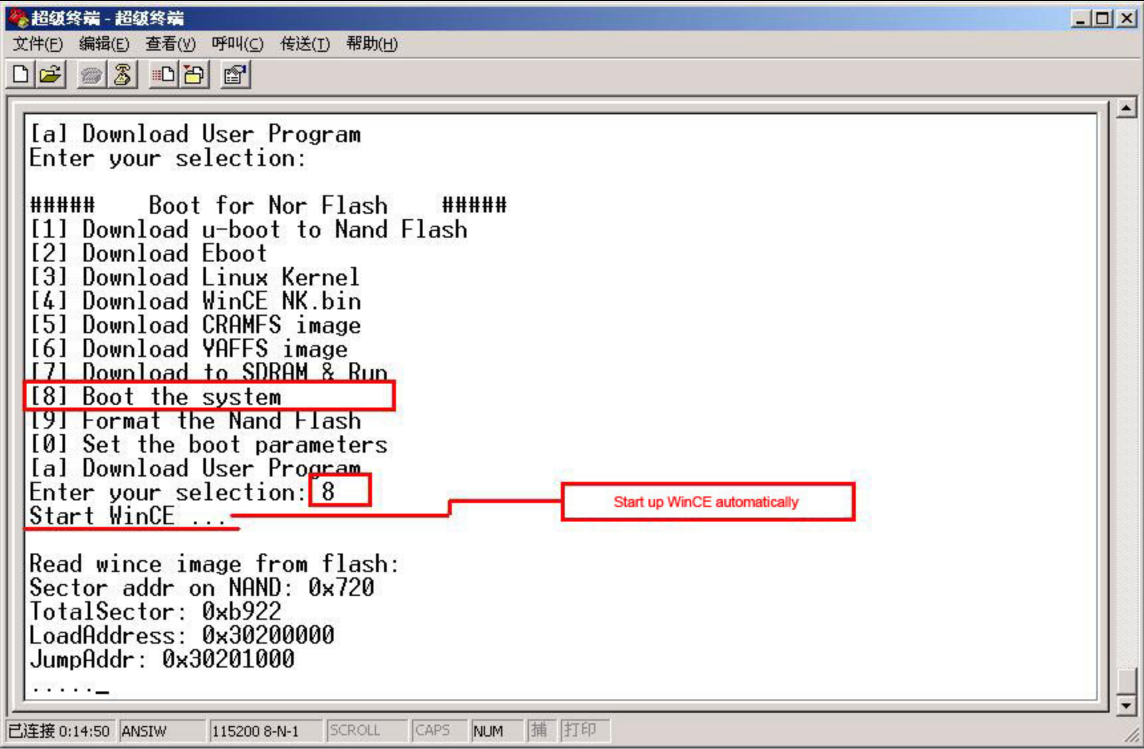


"TQ2440_Test_xxxx.bin" starts to run automatically when download is complete, as shown in the following diagram:



Step 8, enter "8": Boot the system, as shown in the following diagram:

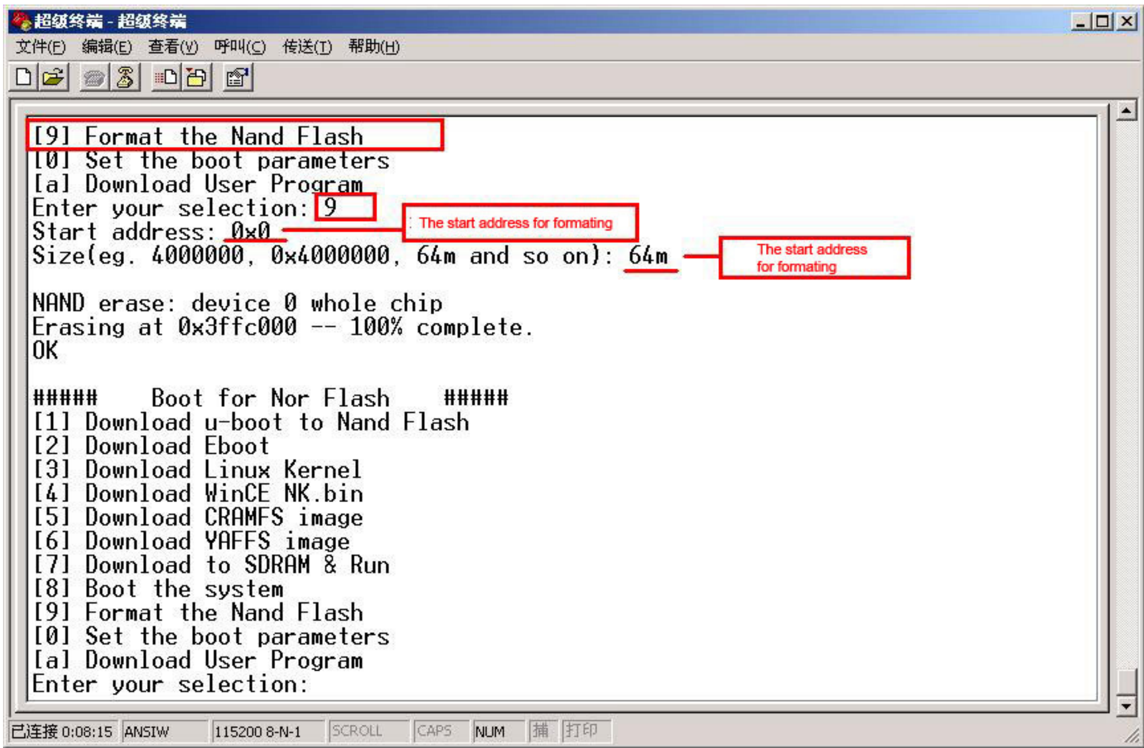
The upper diagram displays Linux start-up, which is similar to WinCE start-up:



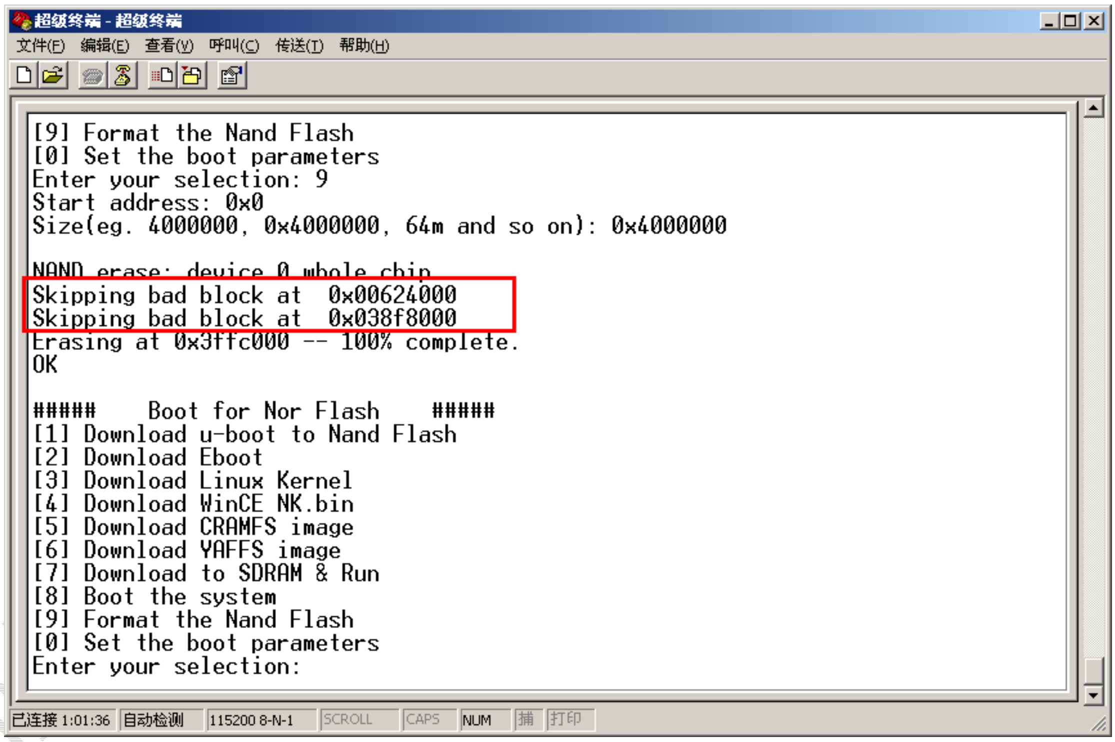
Step 9, enter “9”: Format the Nand Flash (be cautious to use this funtion), as shown in the following diagram:

(this option is corresponding to format Nand Flash)

The maxium formating length is 4000000， namely 64MB Nand Flash.



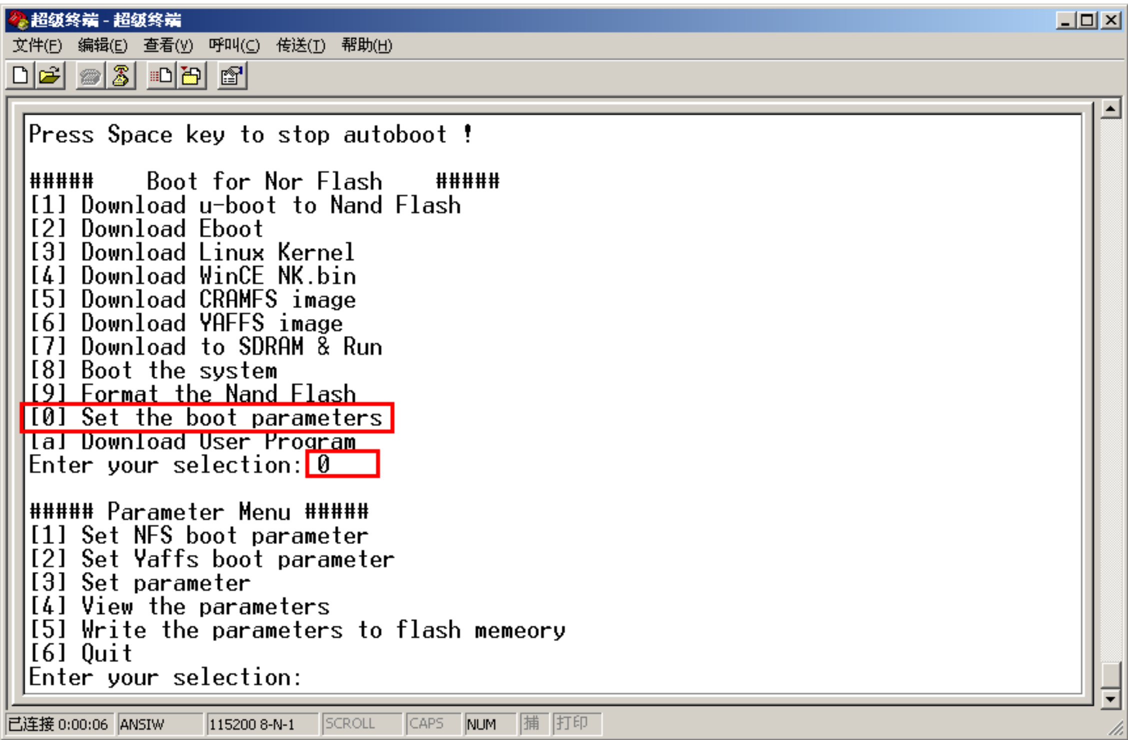
If there are bad blocks in Nand Flash, prompts will appear in the interface during formating, as shown in the following diagram:



(caution: All information preserved in Nand Flash will be wiped up when formatting. Please backup your data before formatting.

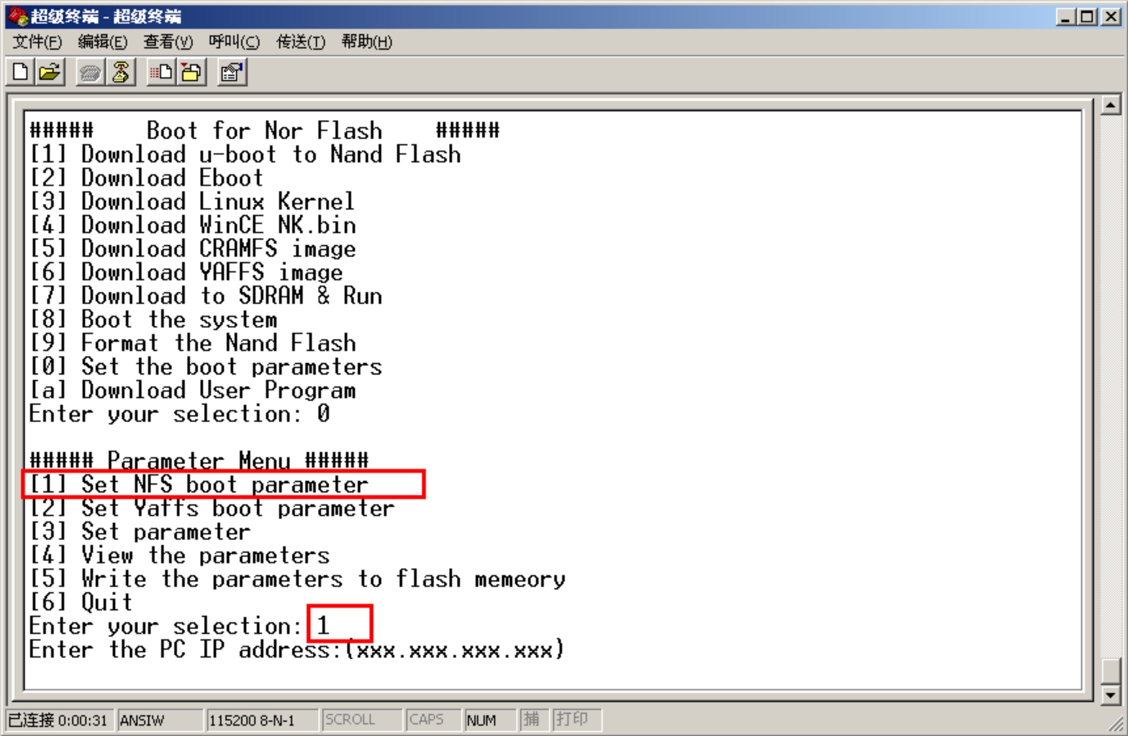
Bad block is always caused by frequent reading, writing or burning Nand Flash. If you often encouter operational errors, it is suggested to format Nand Flash which may recover some bad blocks.)

Step 10, enter “0”: Set the boot parameters, as shown in the following diagram:



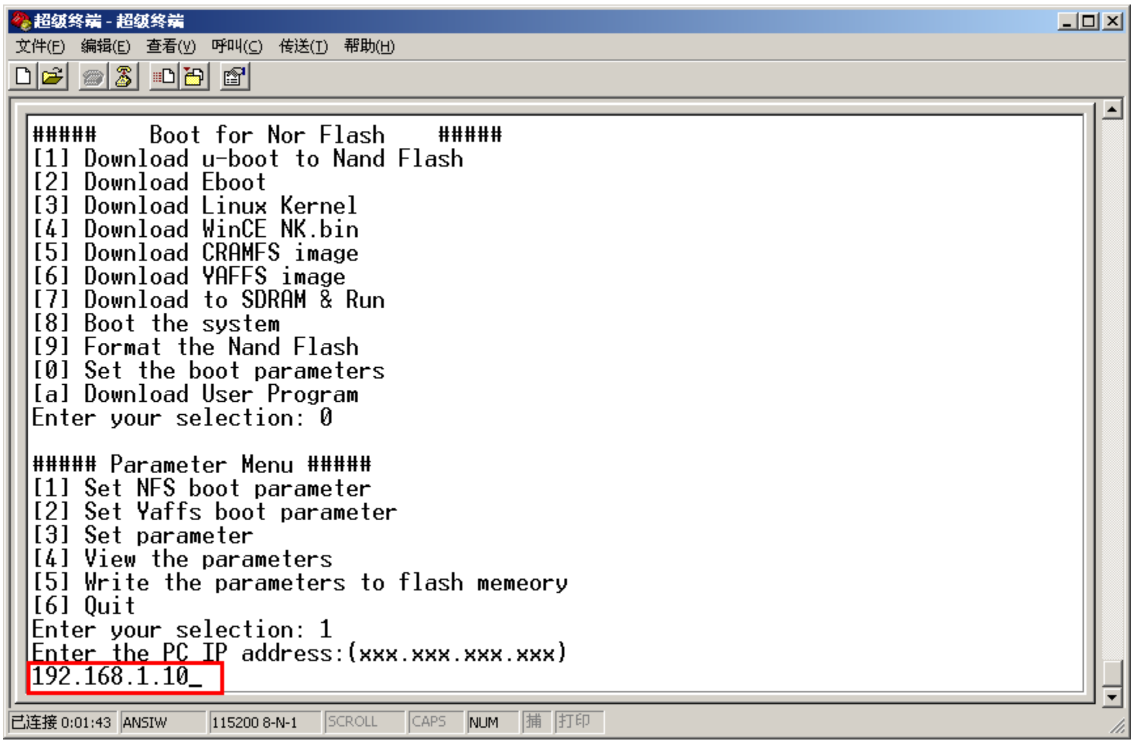
Enter “1”: Set NFS boot parameter, as shown in the following diagram:

(caution: You can also enter “3” to set parameter manually)



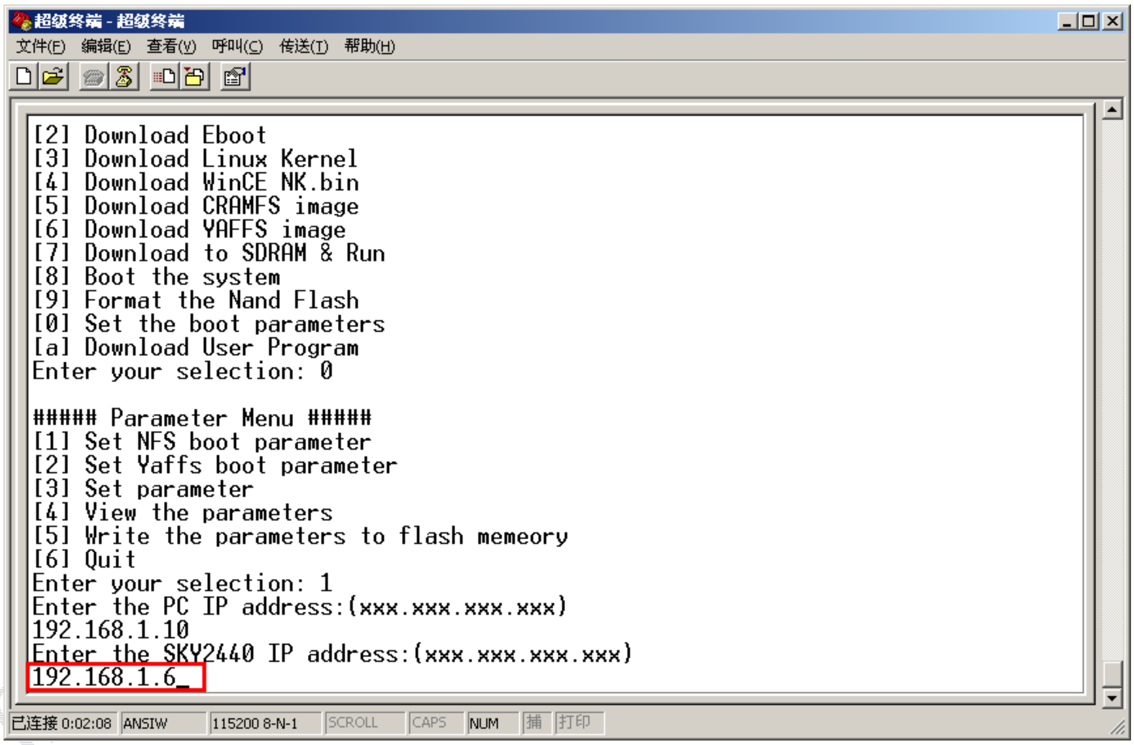
```
##### Boot for Nor Flash #####
[1] Download u-boot to Nand Flash
[2] Download Eboot
[3] Download Linux Kernel
[4] Download WinCE NK.bin
[5] Download CRAMFS image
[6] Download YAFFS image
[7] Download to SDRAM & Run
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program
Enter your selection: 0

##### Parameter Menu #####
[1] Set NFS boot parameter
[2] Set Yaffs boot parameter
[3] Set parameter
[4] View the parameters
[5] Write the parameters to flash memeory
[6] Quit
Enter your selection: 1
Enter the PC IP address:(xxx.xxx.xxx.xxx)
```

Input IP address of PC (we use 192.168.1.10 here)(caution: What you input here is exactly the IP address of Linux on PC. If your Linux is running in virtual machine, input the IP of Linux in virtual machine):

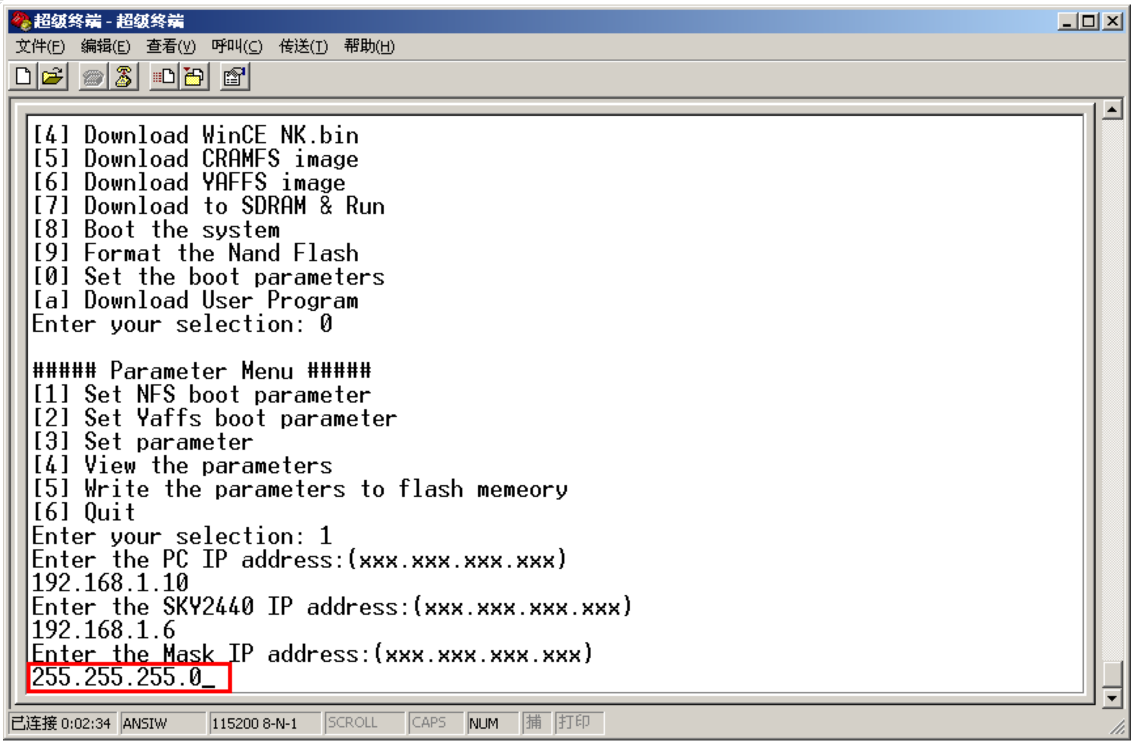


```
##### Boot for Nor Flash #####
[1] Download u-boot to Nand Flash
[2] Download Eboot
[3] Download Linux Kernel
[4] Download WinCE NK.bin
[5] Download CRAMFS image
[6] Download YAFFS image
[7] Download to SDRAM & Run
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program
Enter your selection: 0

##### Parameter Menu #####
[1] Set NFS boot parameter
[2] Set Yaffs boot parameter
[3] Set parameter
[4] View the parameters
[5] Write the parameters to flash memeory
[6] Quit
Enter your selection: 1
Enter the PC IP address:(xxx.xxx.xxx.xxx)
192.168.1.10_
```

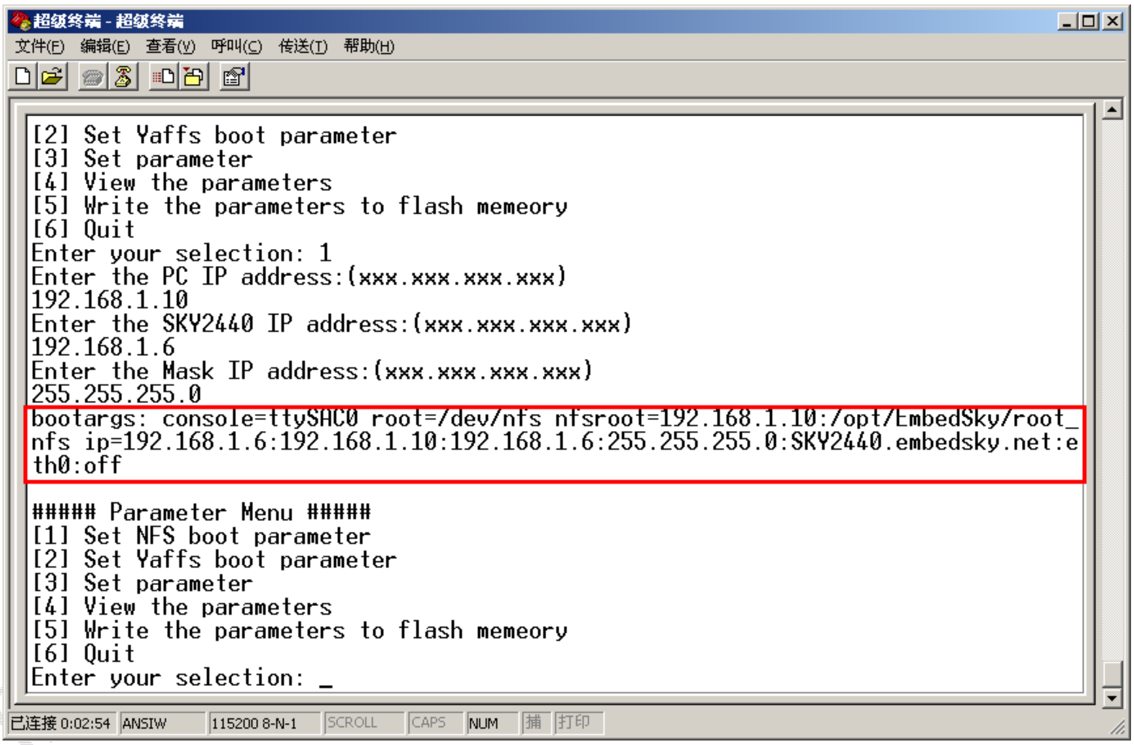
Input IP address of platform (we use 192.168.1.6 here.)(caution: The IP address input here should be same as the IP set in NFS , otherwise NFS can not start up. The default IP set in NFS is 192.168.1.6, you can modify the file “etc/init.d/rcS” to reset IP address.):



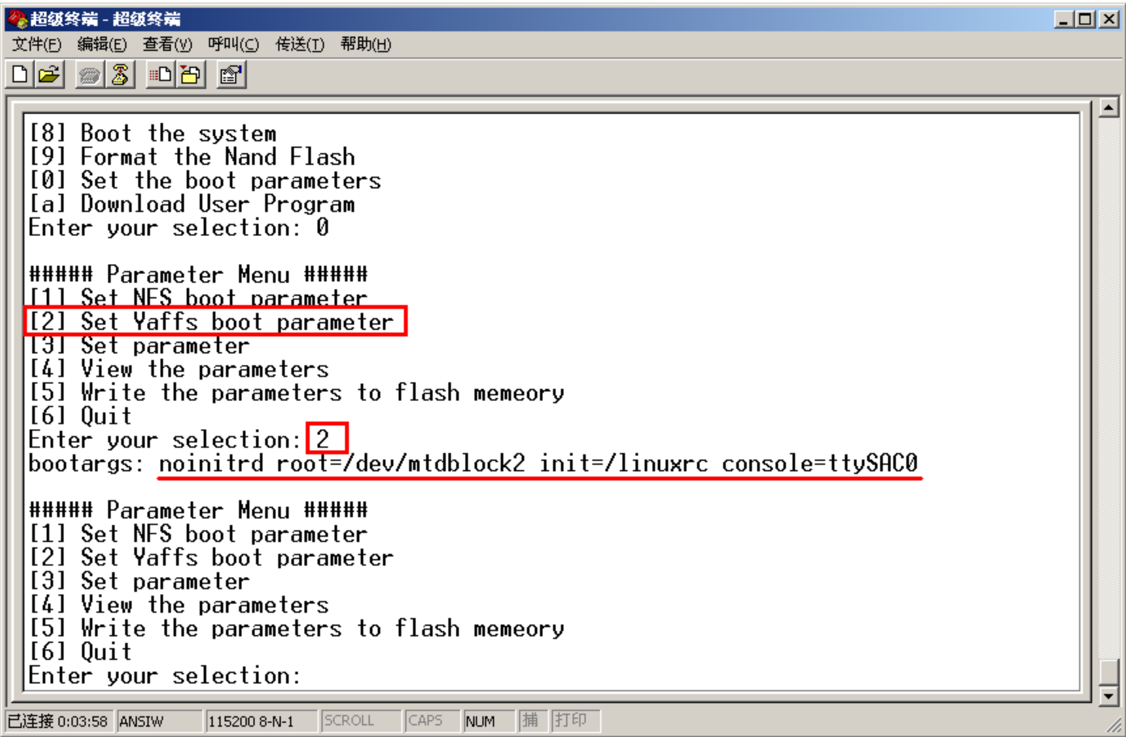
Enter return-key and enter the Mask IP address (typically 255.255.255.0):



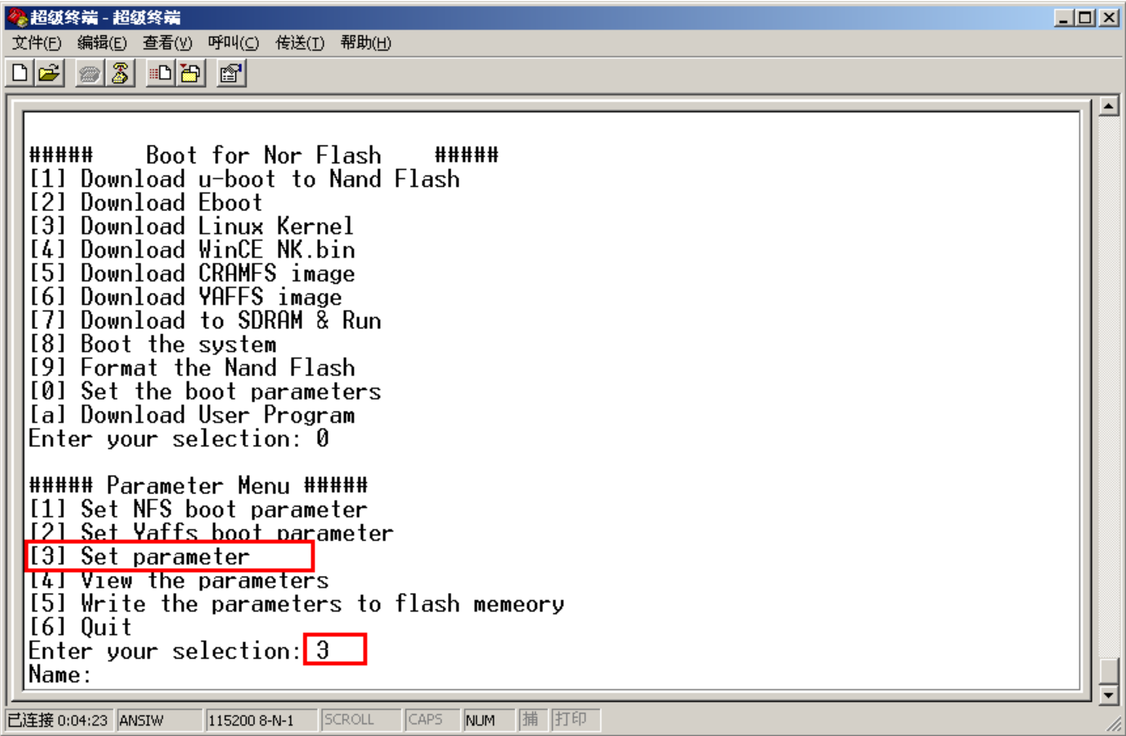
Enter return-key to continue:



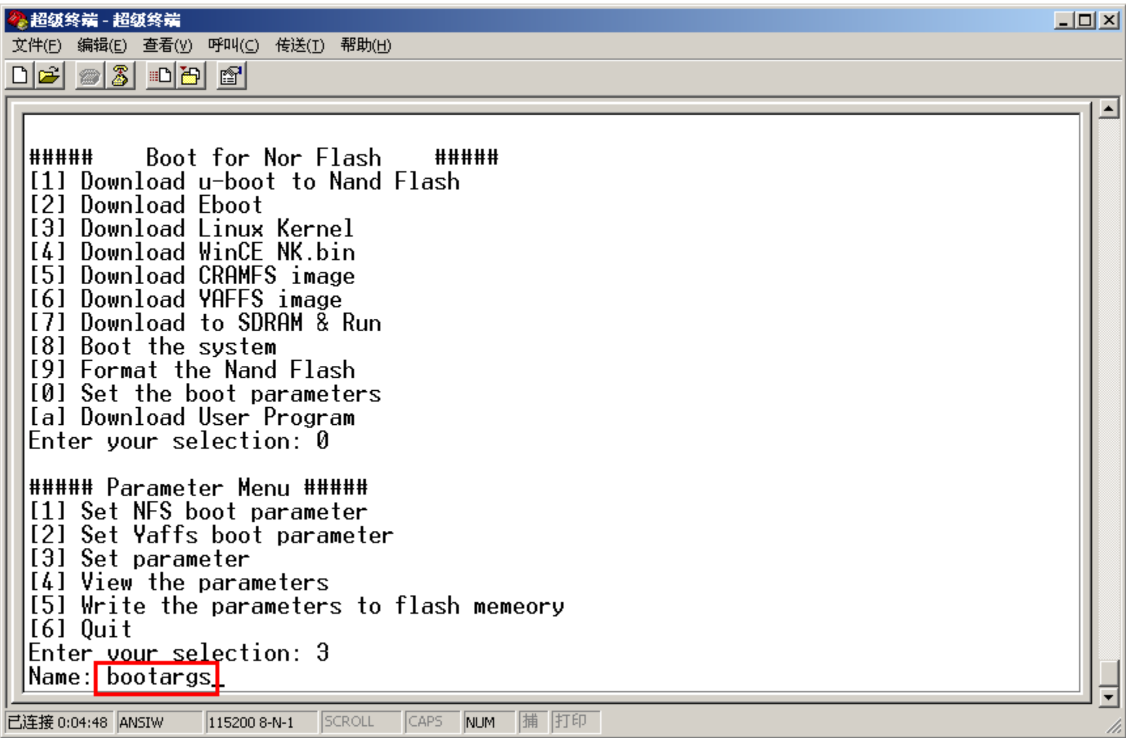
Enter “2”: Set Yaffs boot parameter, as shown in the following diagram:



Enter “3”: Set parameter, as shown in the following diagram:



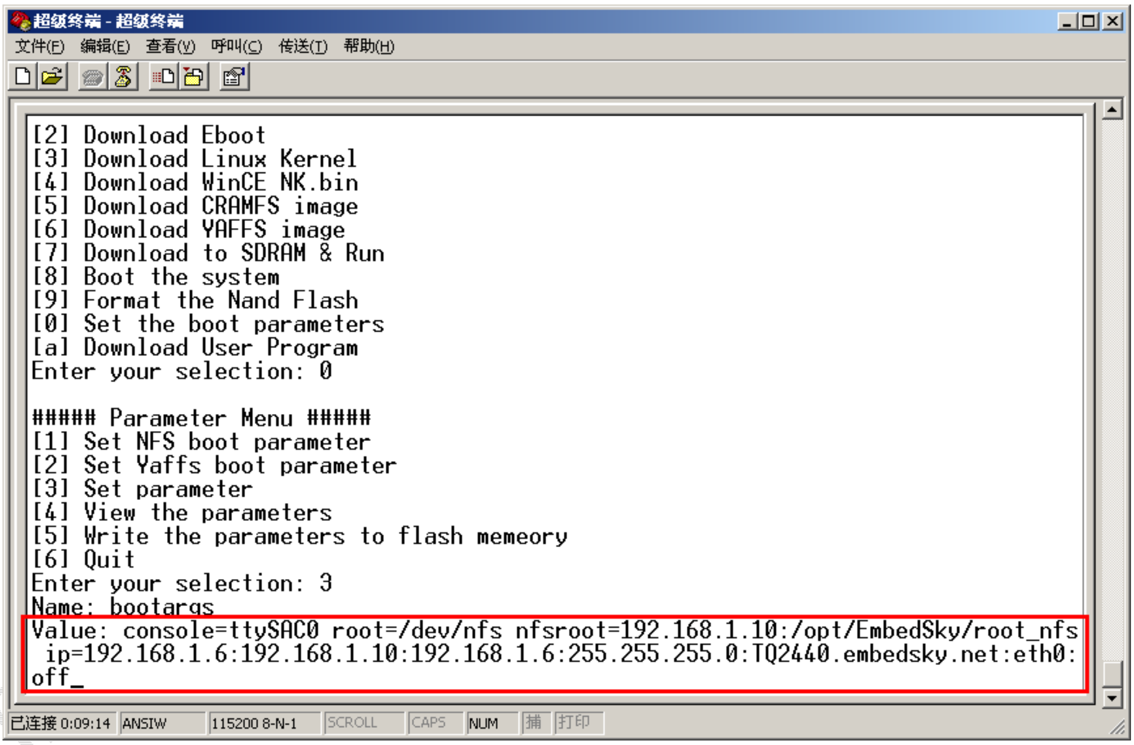
Step 10.3.1, input the parameter “bootargs” (this parameter is transferred to Linux by u-boot when start-up). Enter “回车” to continue: (we use NFS start-up configuration for example here)



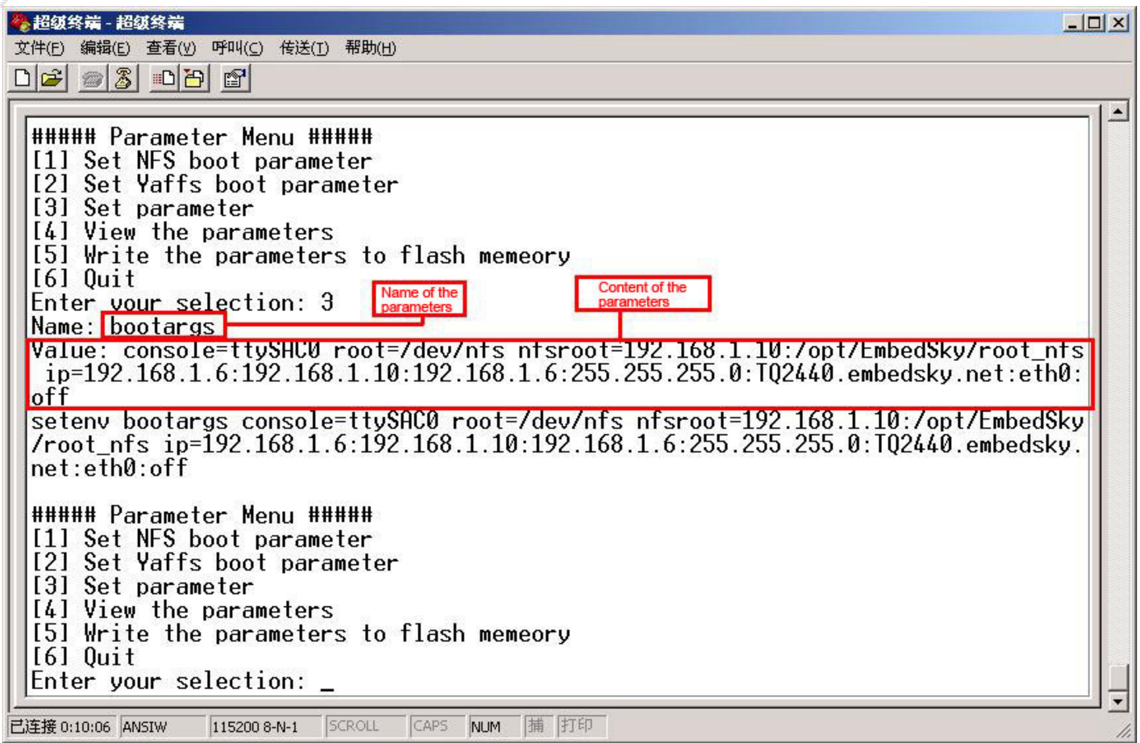
Step 10.3.2, input NFS boot parameter

“console=ttySAC0 root=/dev/nfs nfsroot=192.168.1.10:/opt/EmbedSky/root_nfs ip=192.168.1.6:192.168.1.10:192.168.1.6:255.255.255.0:TQ2440.embedsky.net:eth0:off”,

Enter return-key to continue (not include the quotation marks)

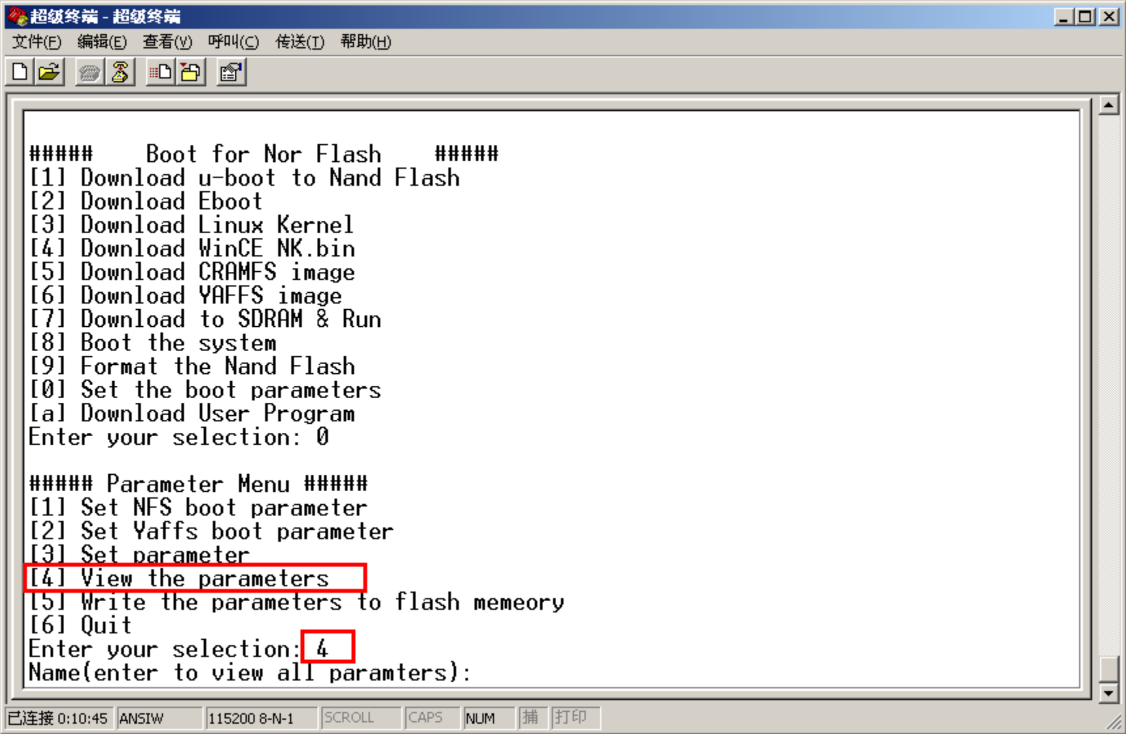


The following diagram briefly illustrates the parameter:

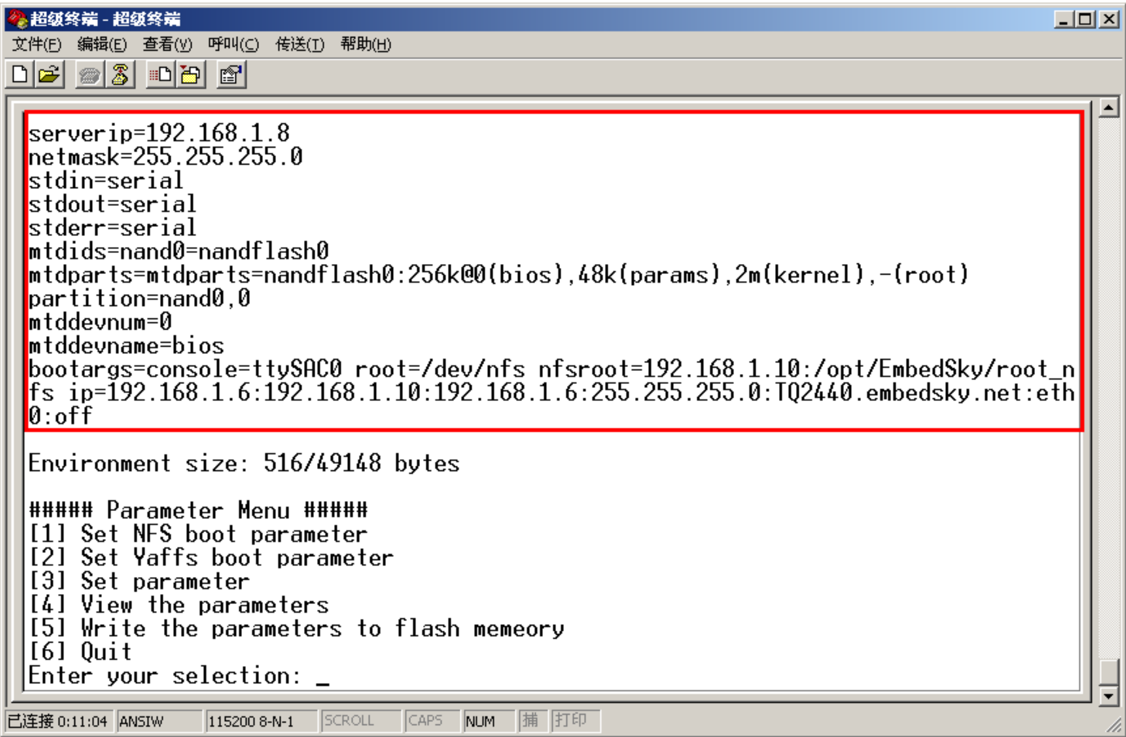


Enter "5" to save the configuration.

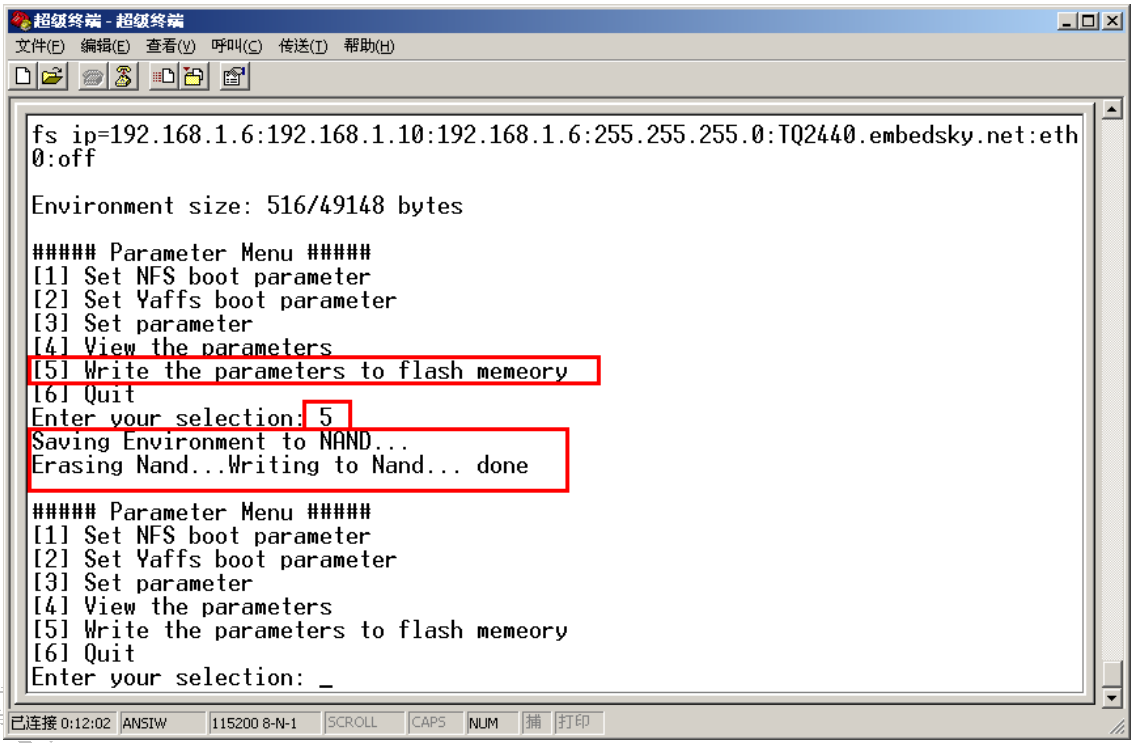
Enter "4": View the parameters. The parameter list appears in the following diagram:



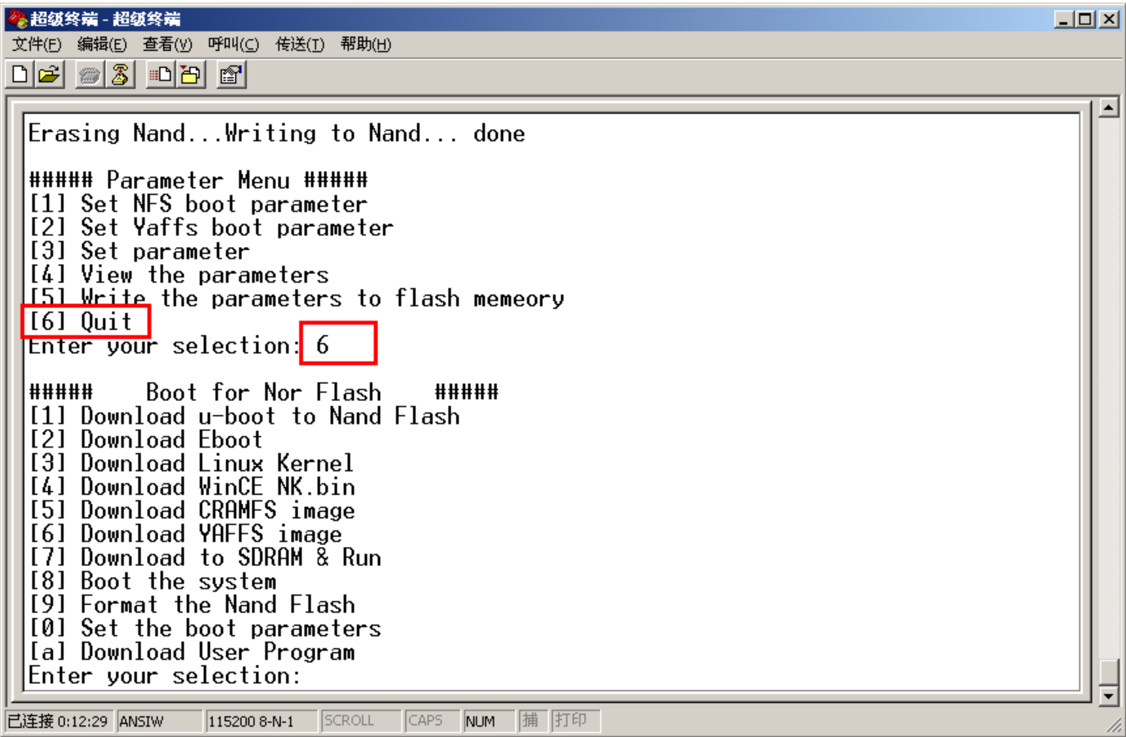
The environment variable list is shown in the following diagram:



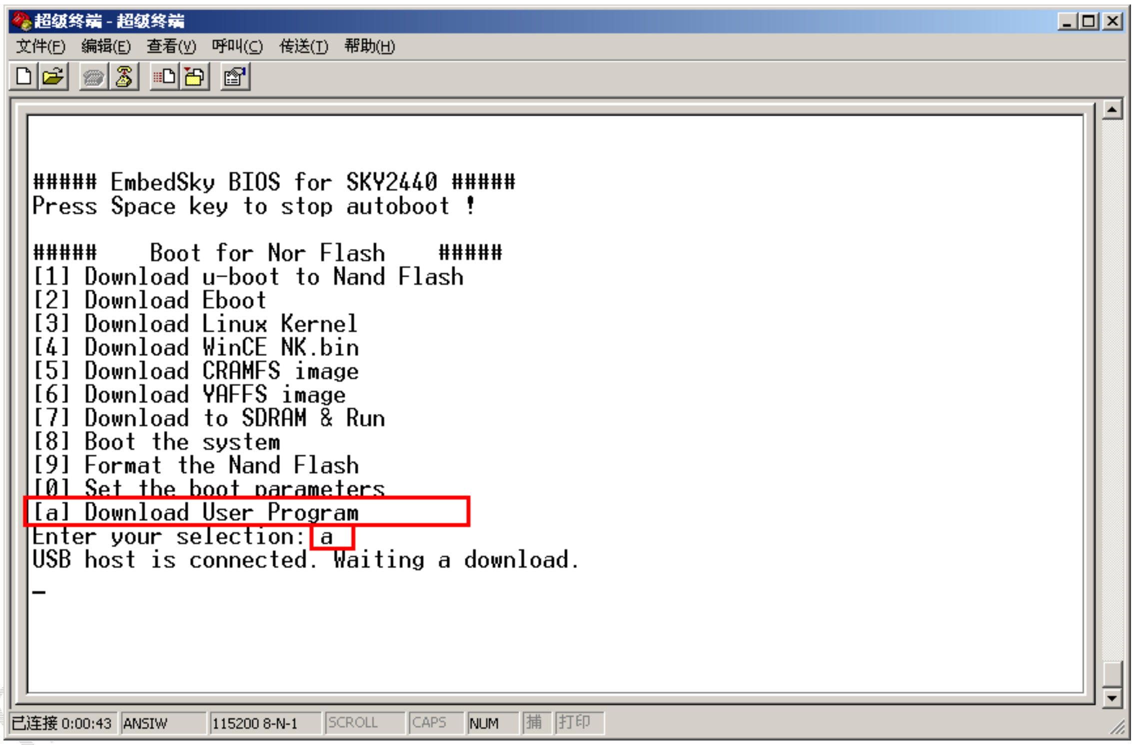
Enter “5”: Write the parameters to flash memory, as shown in the following diagram:



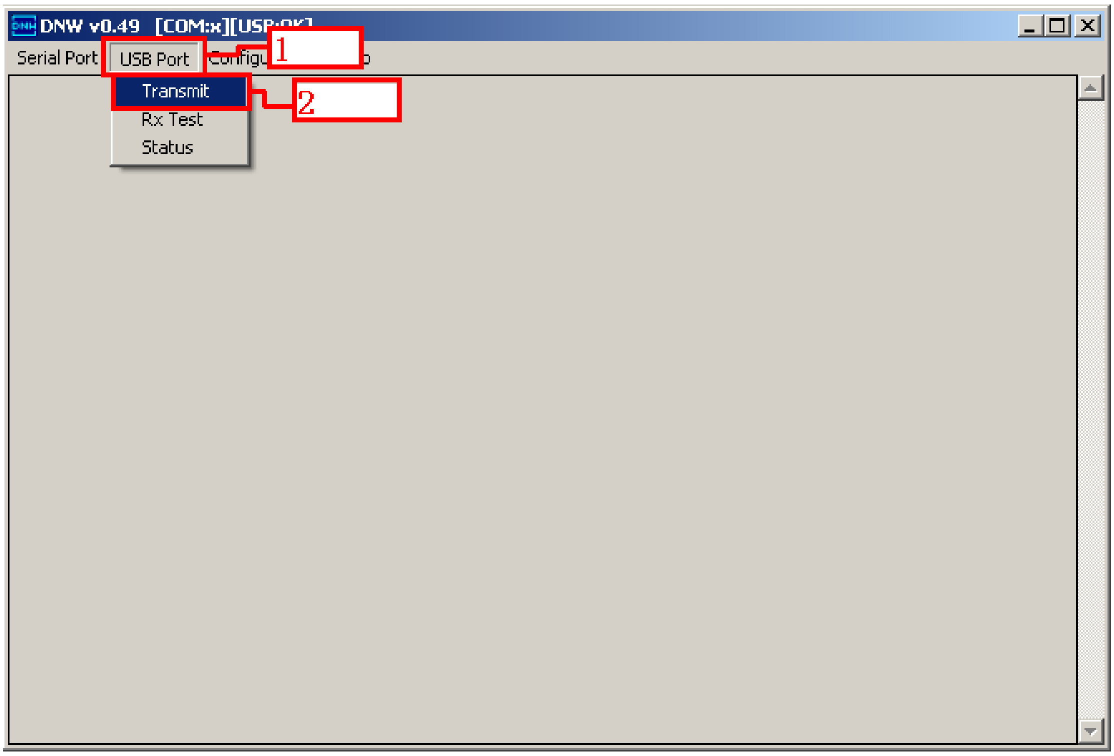
Enter “6”: Quit to the upper directory, as shown in the following diagram:



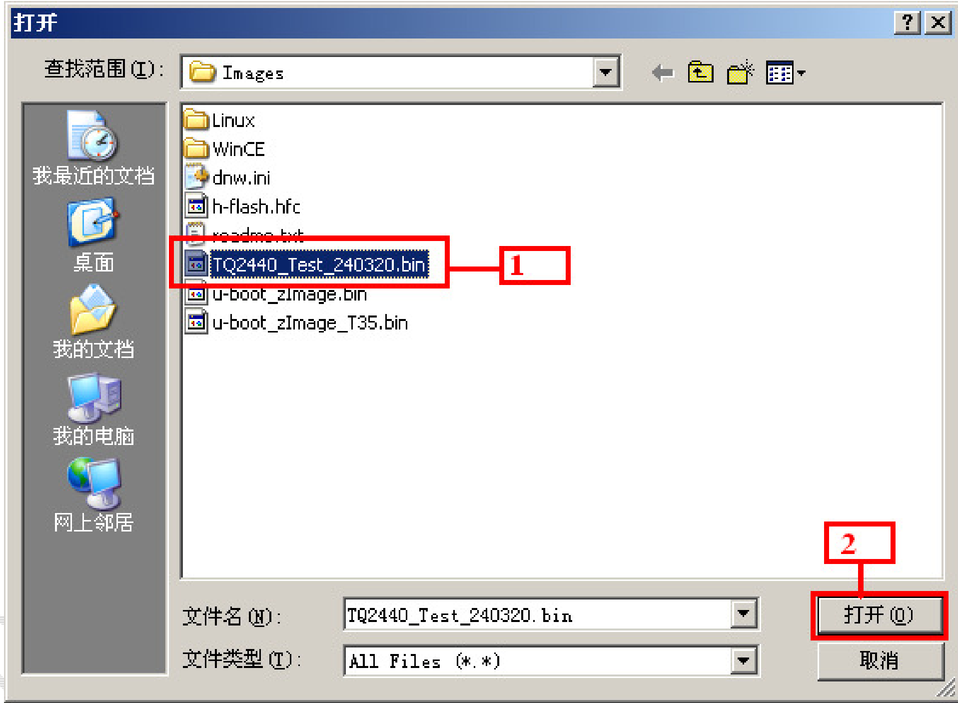
Step11, enter “a”: Download User Program to Nand Flash, starting from block 0. For example, write non-OS testing program to Nand Flash:



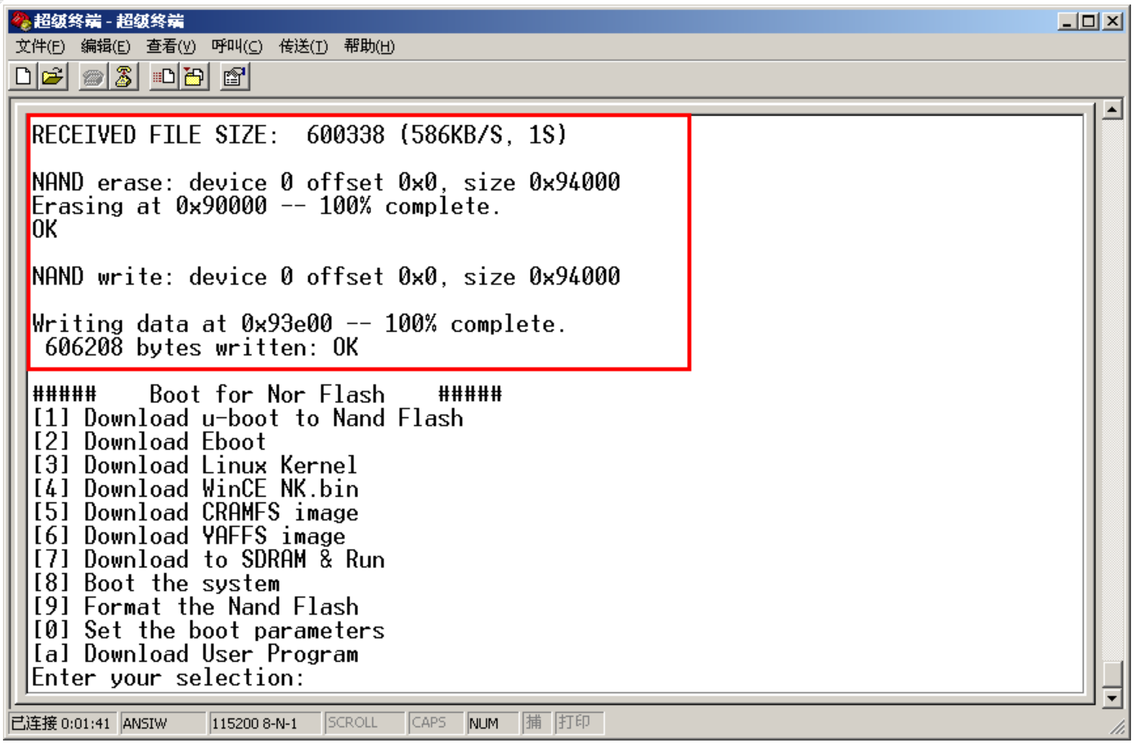
Open the software DNW when the upper sentence “USB hosting is connected，Waiting a download.” appears (you need to wait about 5 minutes for the sentence). Click “Transmit” in “USB Port” menu:



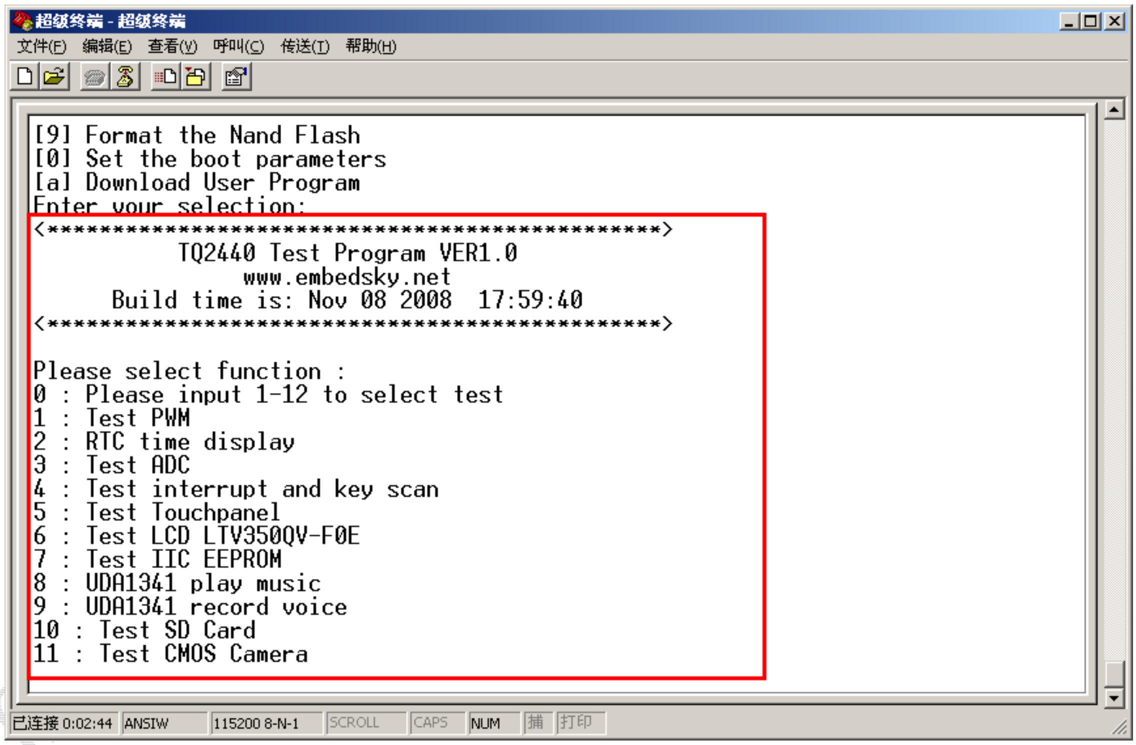
Locate “TQ2440_Test_xxxx.bin” under the directory “Images” in CD-ROM (xxxx represents LCD resolution). Click “打开” to continue:



"TQ2440_Test_xxxx.bin" runs automatically when download is complete, as shown in the following diagram:



The platform starts up from Nand Flash, as shown in the following diagram:

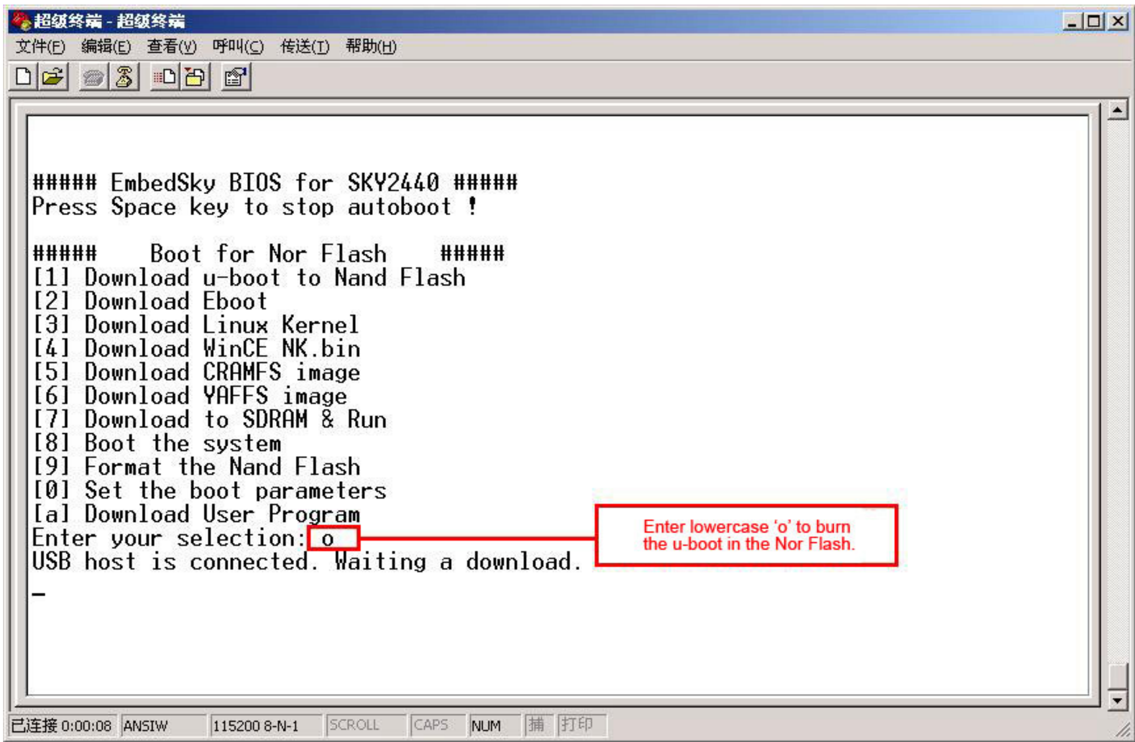


The following 2 operations are hidden operations:

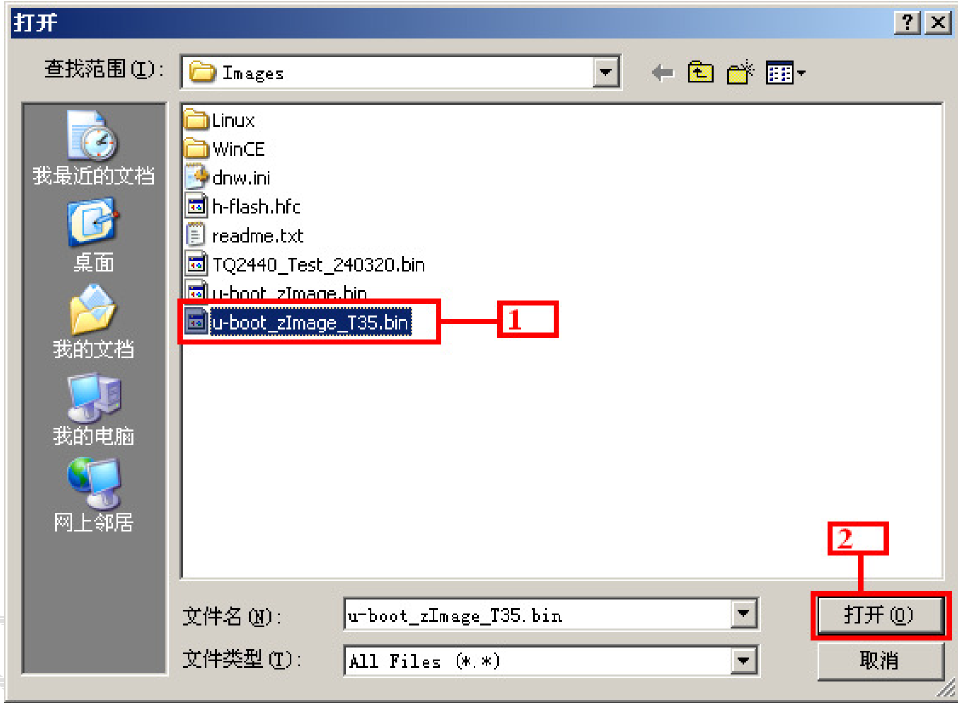
Step12, download u-boot to Nor Flash. This operation needs start-up from Nor Flash.

Caution: Please restart the platform after writing u-boot to Nor Flash if you try to burn other programs. Otherwise system failure might happen.

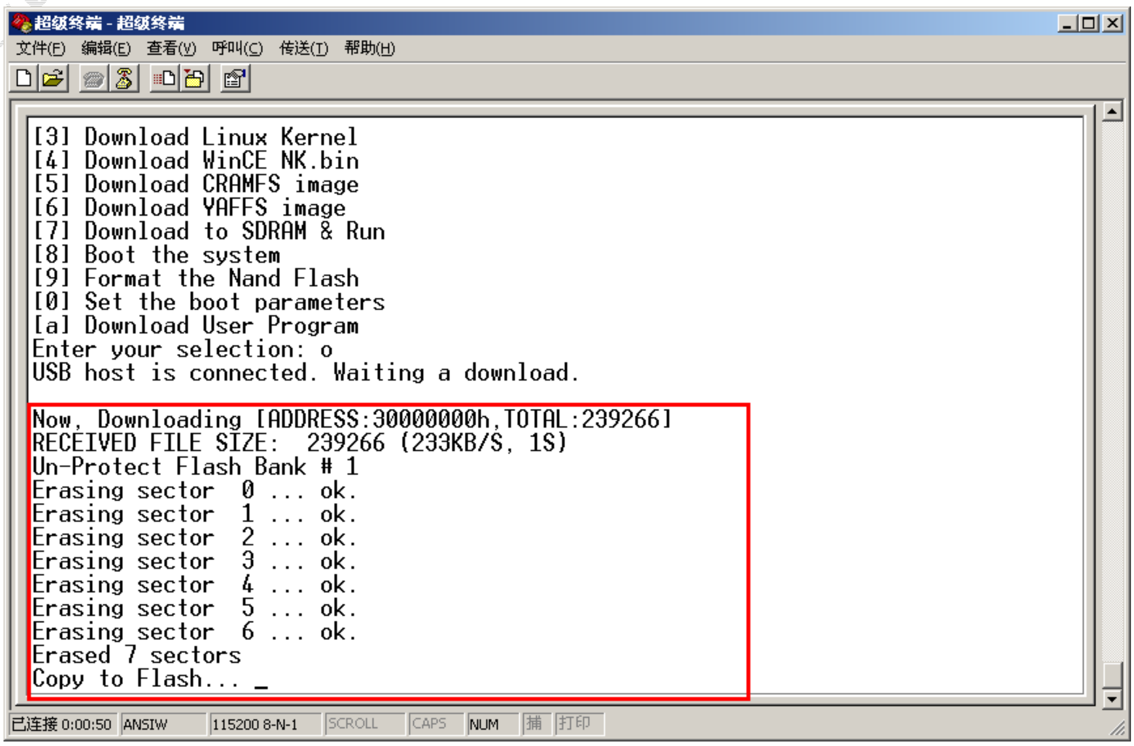
Enter lowercase “o”, as shown in the following diagram:



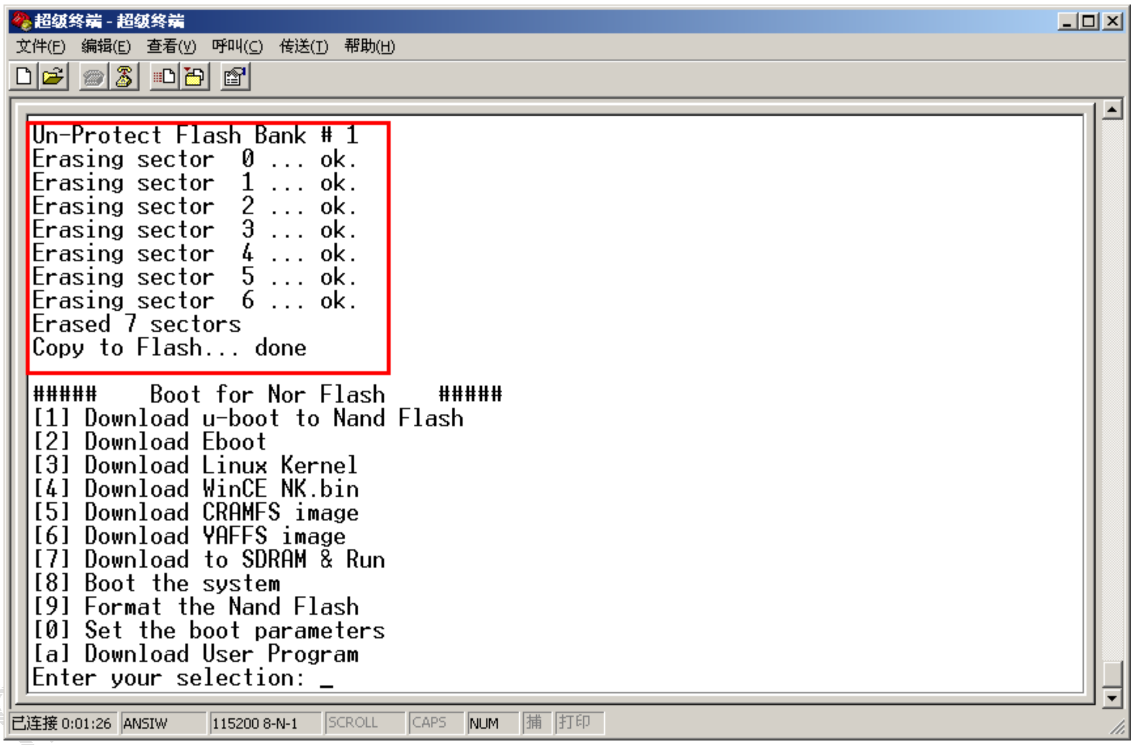
Use software DNW to select the file “u-boot_zImage_xxxx.bin” (xxxx represents LCD type):



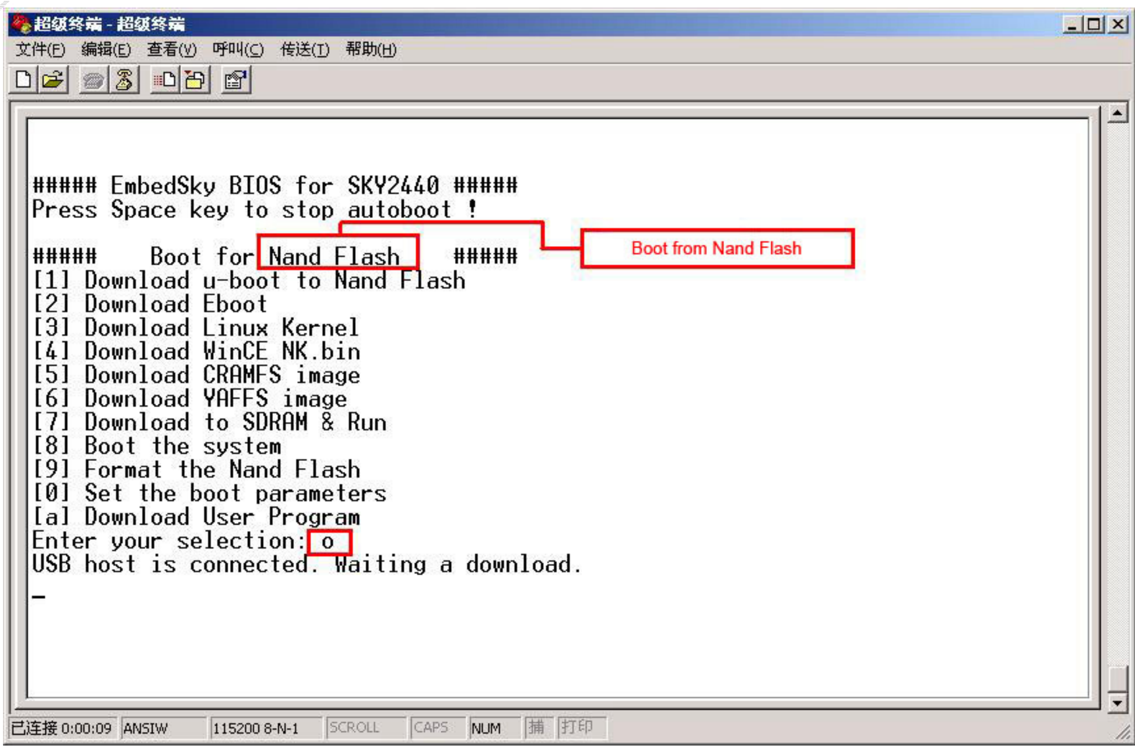
u-boot is download and saved automatically. The following diagram displays the process writing Nor Flash:



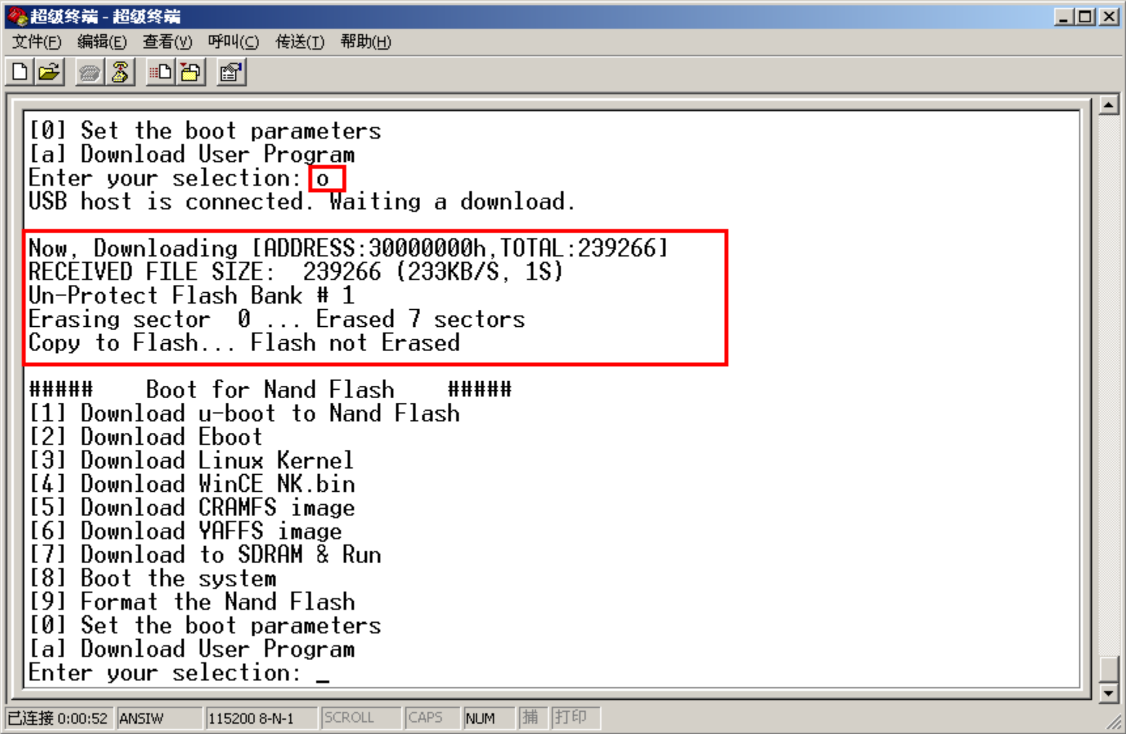
Please reset the platform when writing Nor Flash is finished:



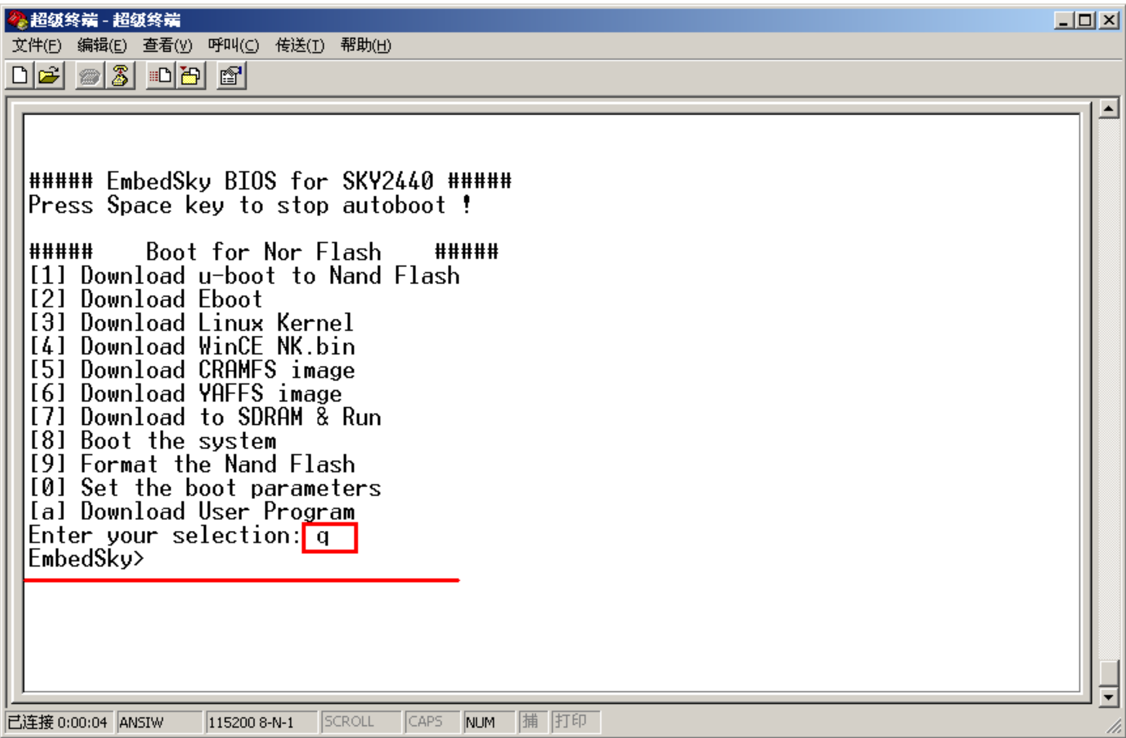
The following diagram displays boot for Nand Flash:



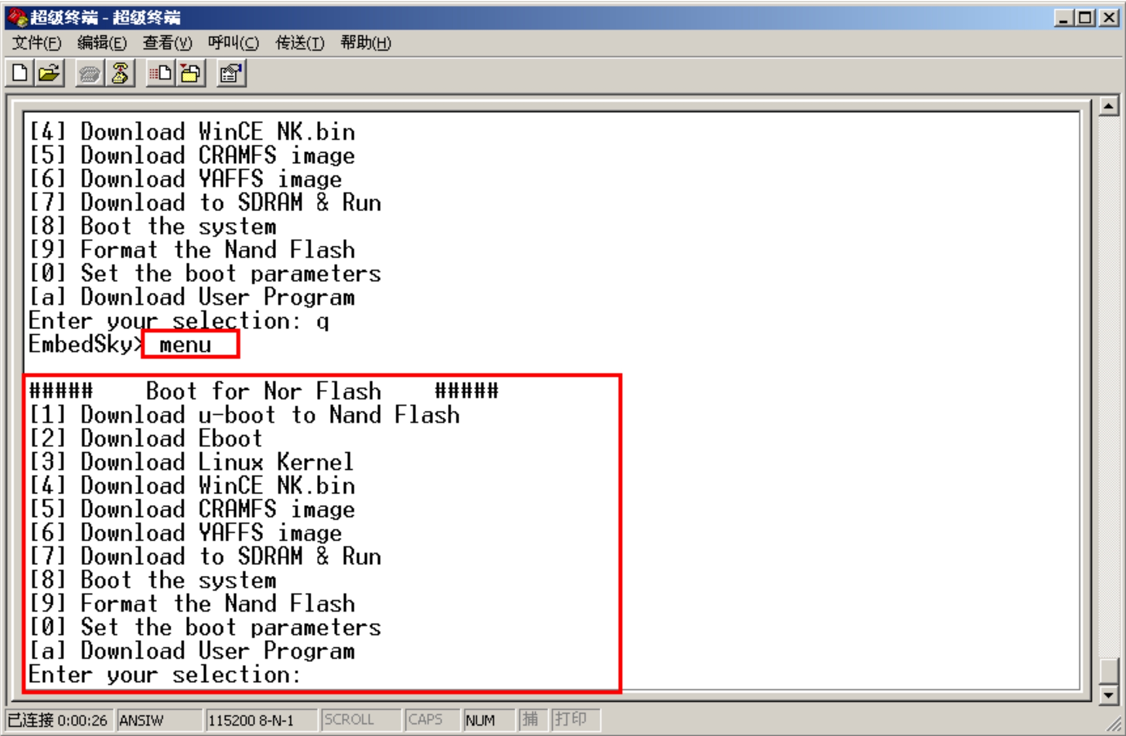
The following diagram displays a failure situation:



Step13, back to u-boot operation interface: enter the key “q” to back to u-boot operation interface, as shown in the following diagram:

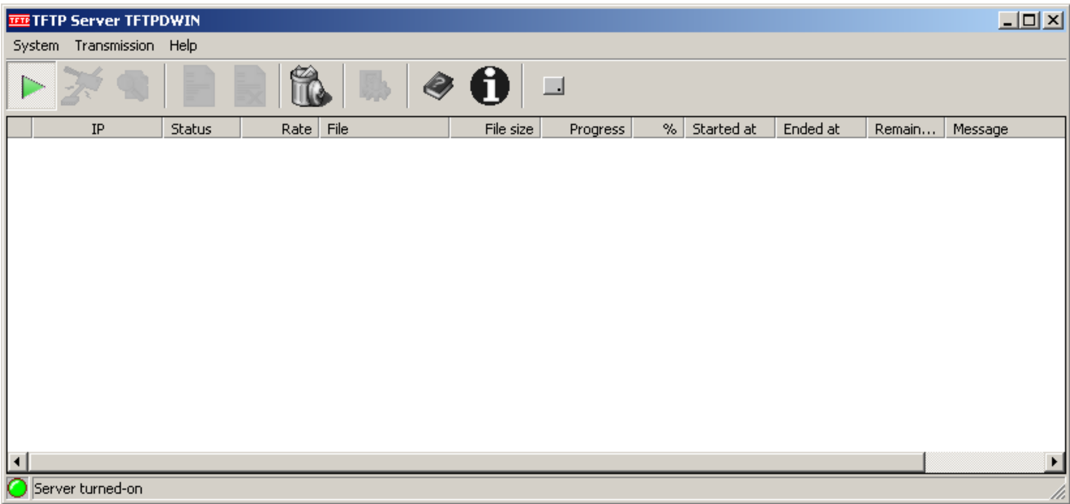


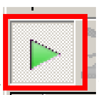
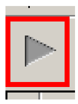
Execute the command “menu” to go back to one-key operation interface:

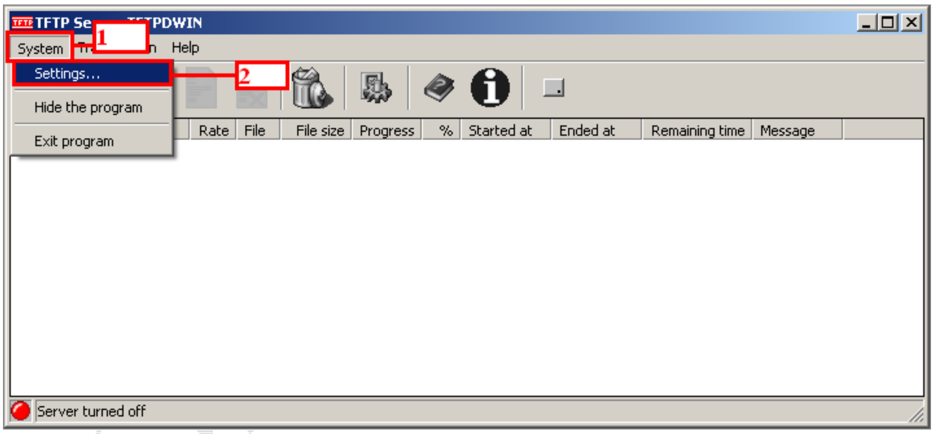


The following contents illustrate the download processs in uboot operation interface:

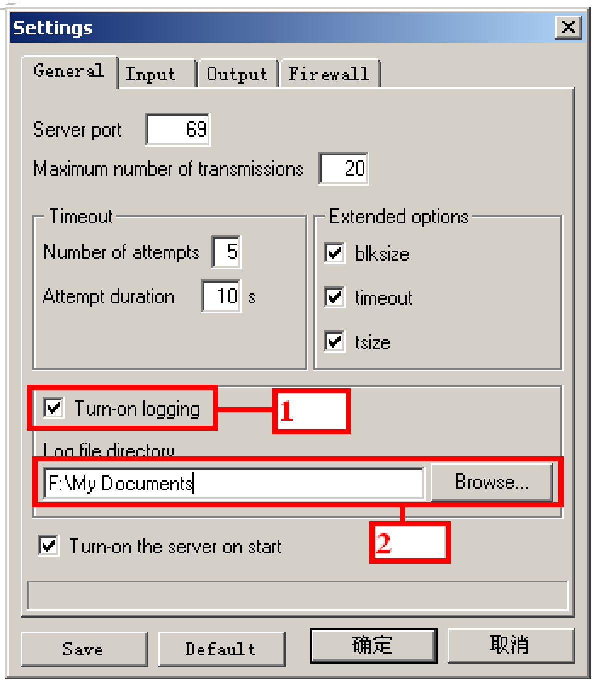
Start TFTP proxy “TFTP Server TFTPDWIN” in Windows:



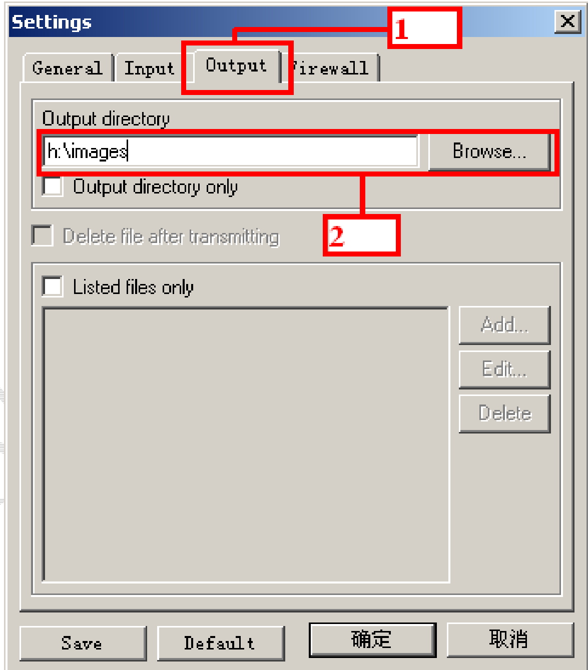
Click the icon “” to stop all the operations of TFTP. The icon turns into “”. Click “System->Setting” to start setting, as shown in the following diagram:



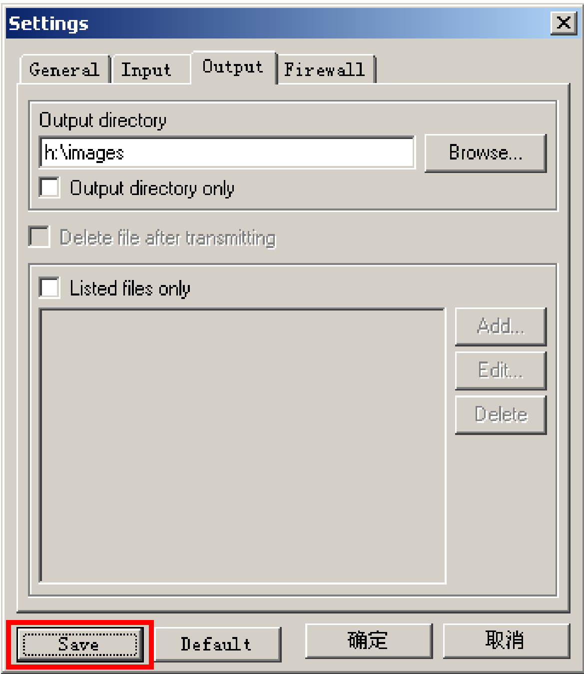
Set the saving path in page “General”. We set “F:\My Documents” here:



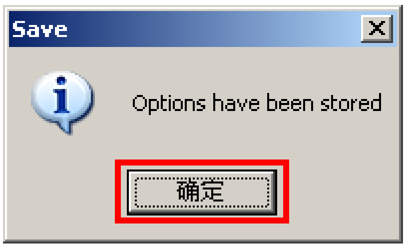
Set output directory in page “Output” (base on your actual situation), as shown in the following diagram:



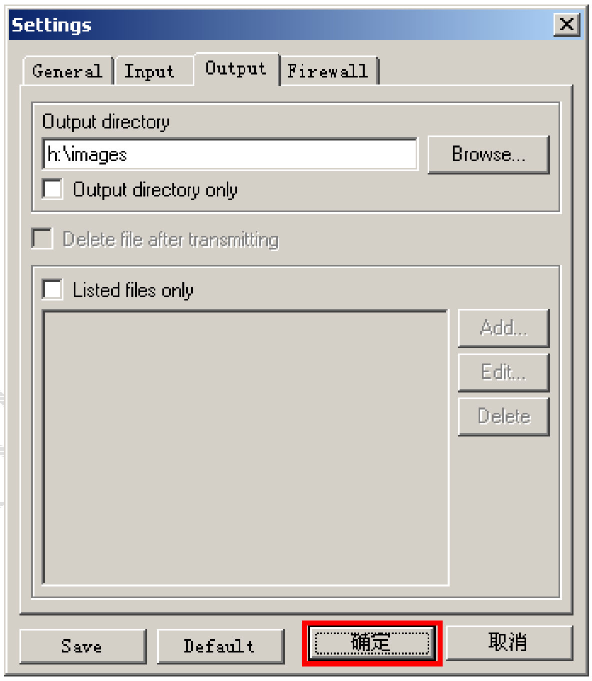
Click “save ” to save your configuration:

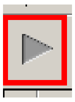
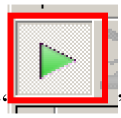


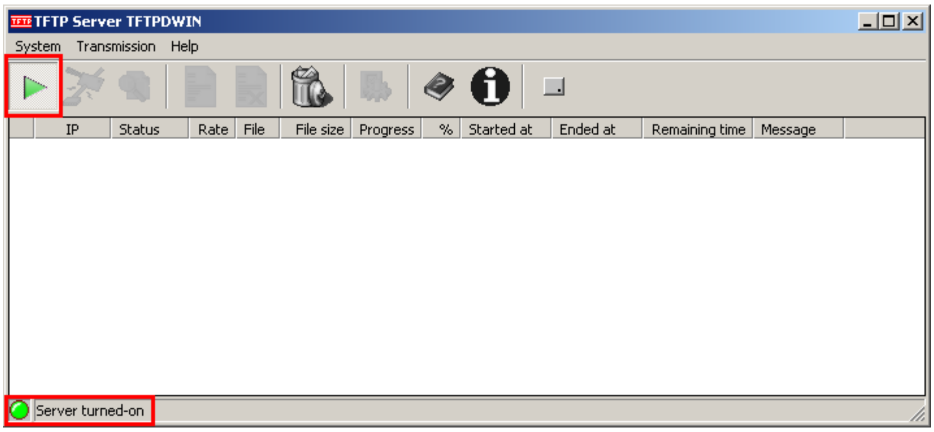
A box “Options have been stored” appears, click “确定” to go back to the upper interface



Click “确定” to complete the setting:

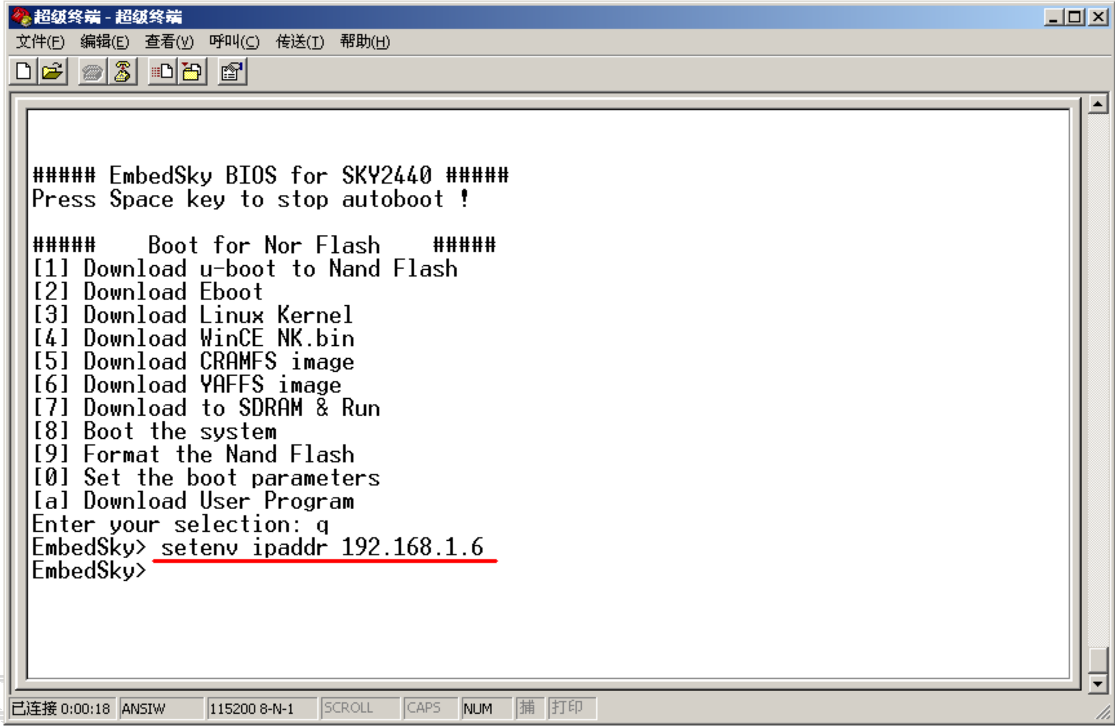


Click the icon "" to start TFTP service, and the icon turns into "":

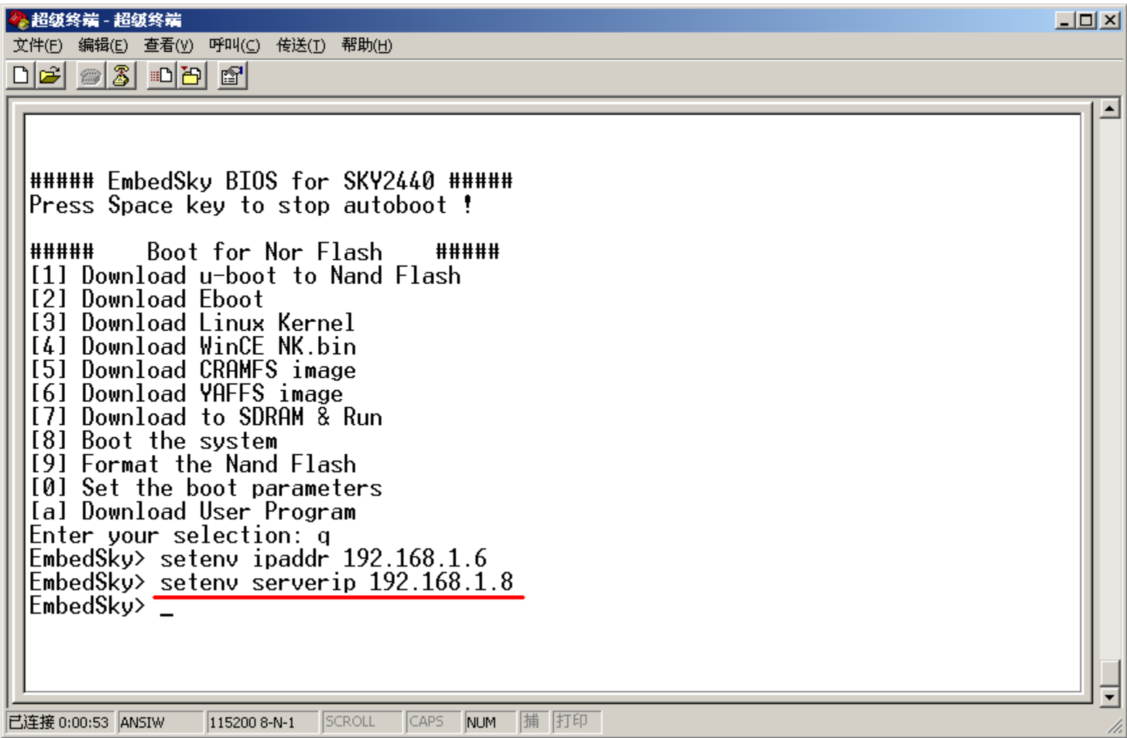


Step13, configure network parameters before using the download function of TFTP (the default IP address set by us probably doesn't fit your need, you can reset it according to the actual situation):

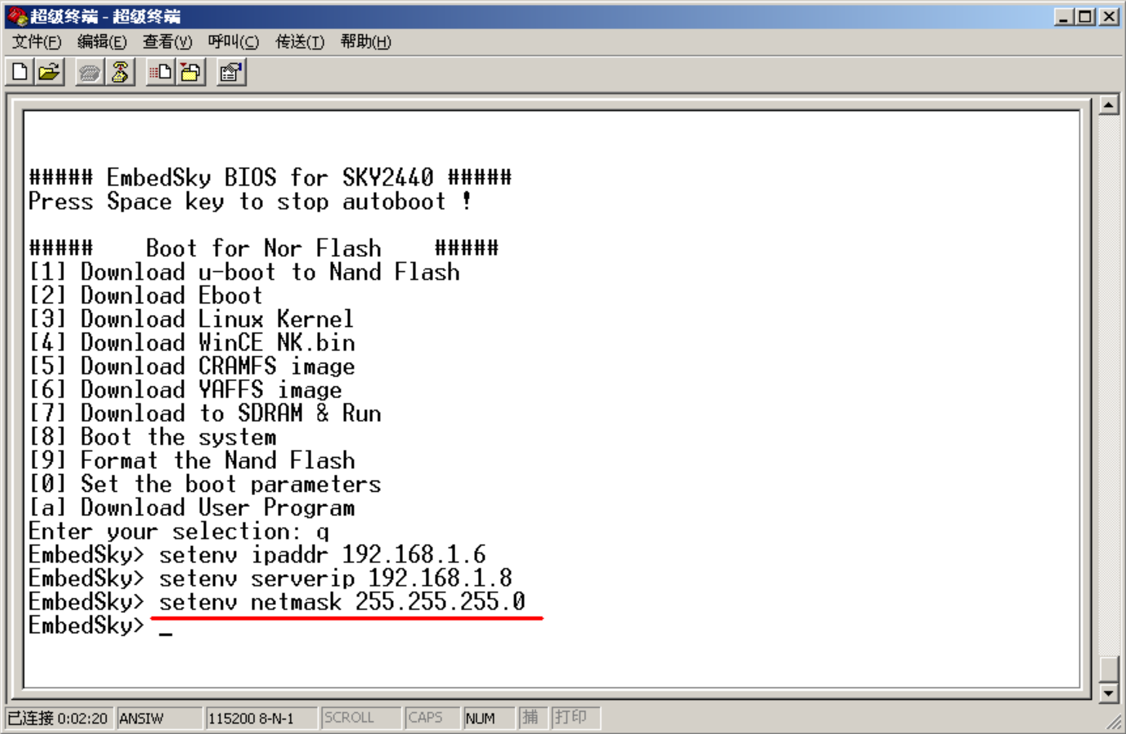
1, set IP address of platform: Execute the command "setenv ipaddr 192.168.1.6", as shown in the following diagram:



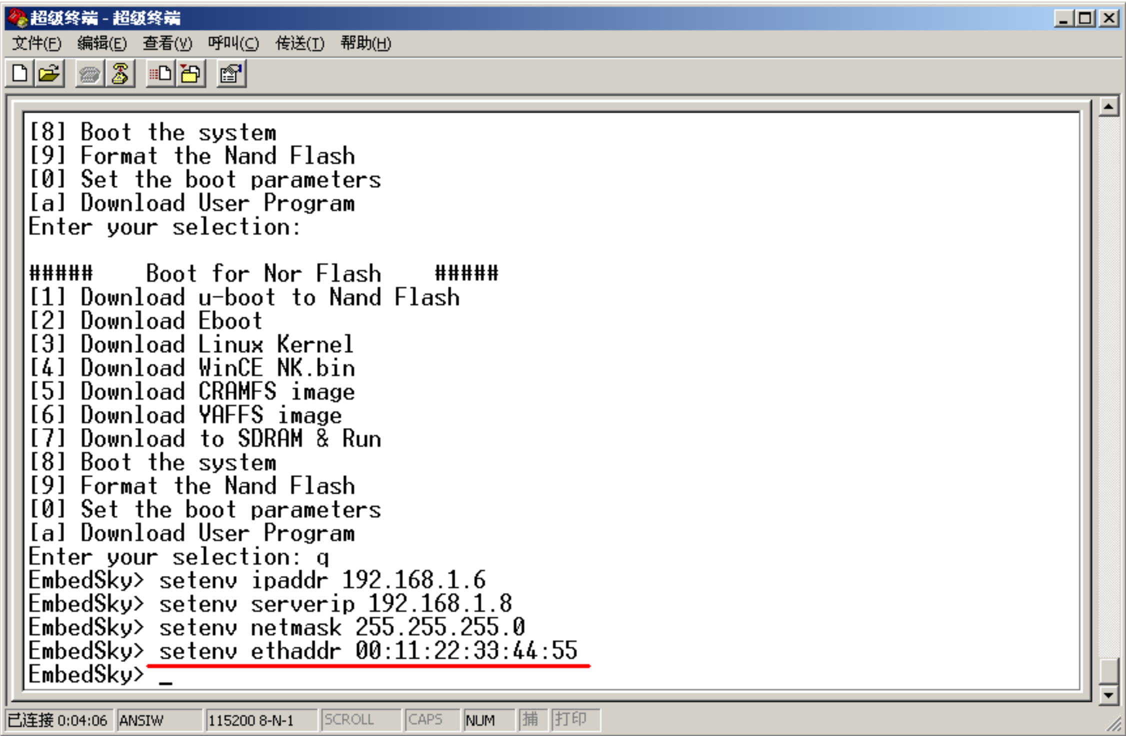
2, set IP address of TFTP server (this IP should be the same as the IP in Windows OS. We use “192.168.1.8” here). Execute the command “setenv serverip 192.168.1.8”, as shown in the following diagram:



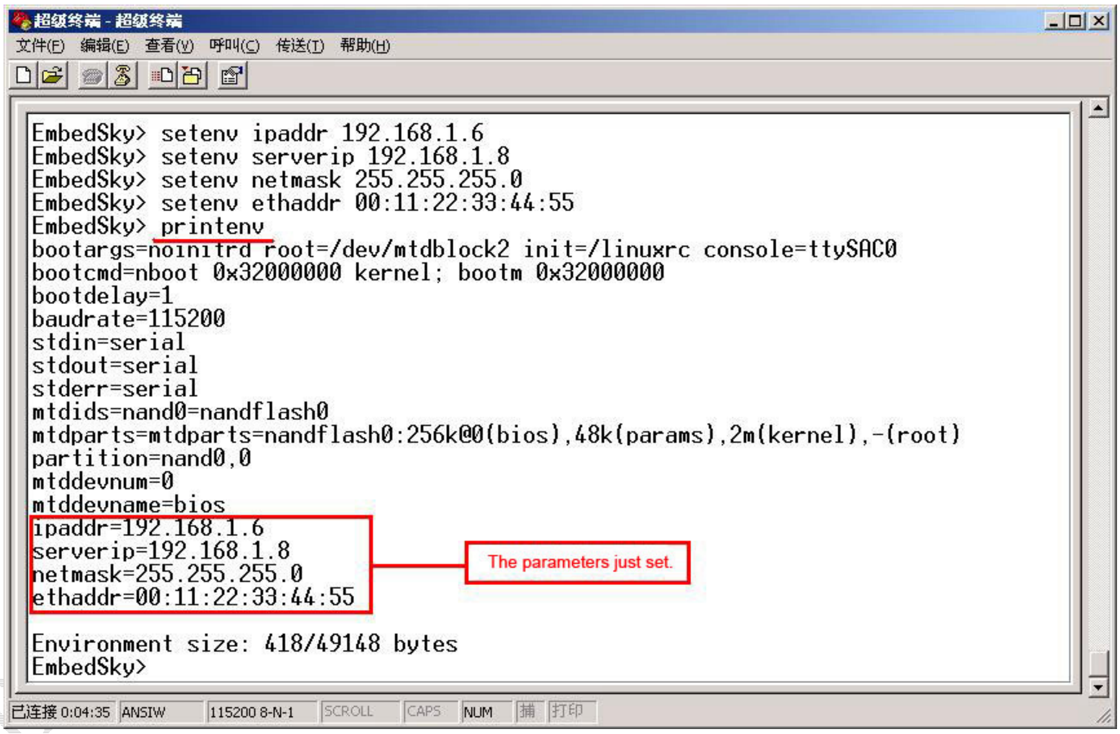
3, set netmask. Execute the command “setenv netmask 255.255.255.0”, as shown in the following diagram:



4, set MAC. Execute the command "setenv ethaddr 0a:1b:2c:3d:4e:5f", as shown in the following diagram:



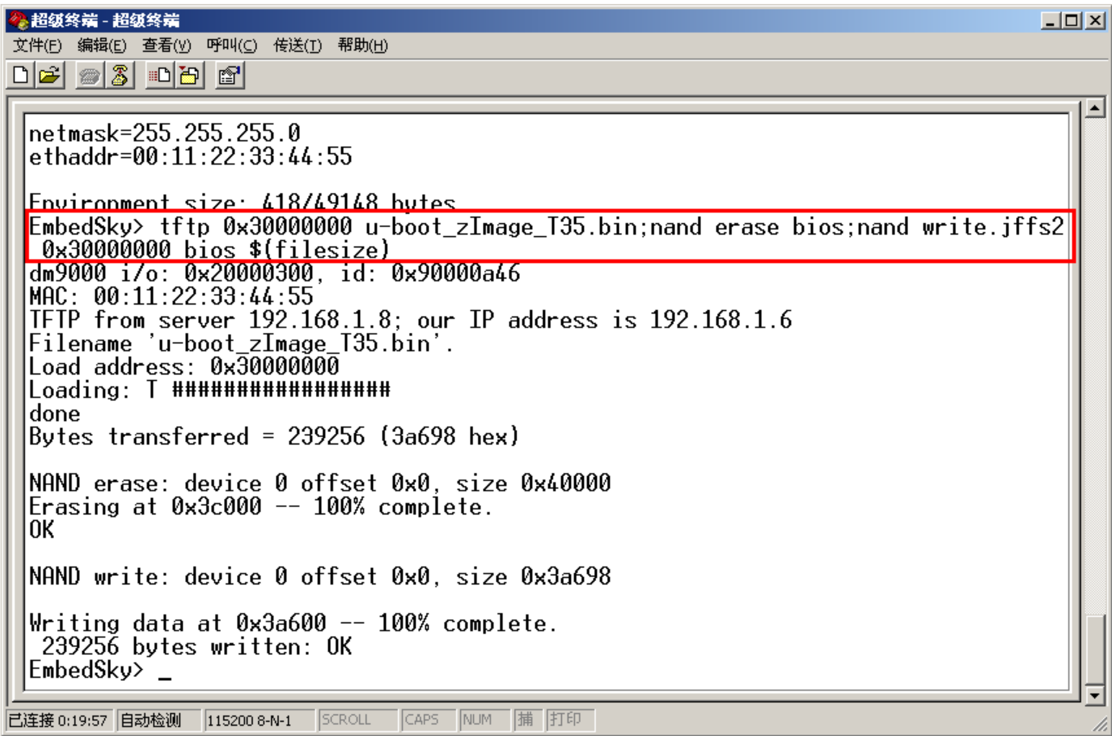
5, check your settings, as shown in the following diagram:



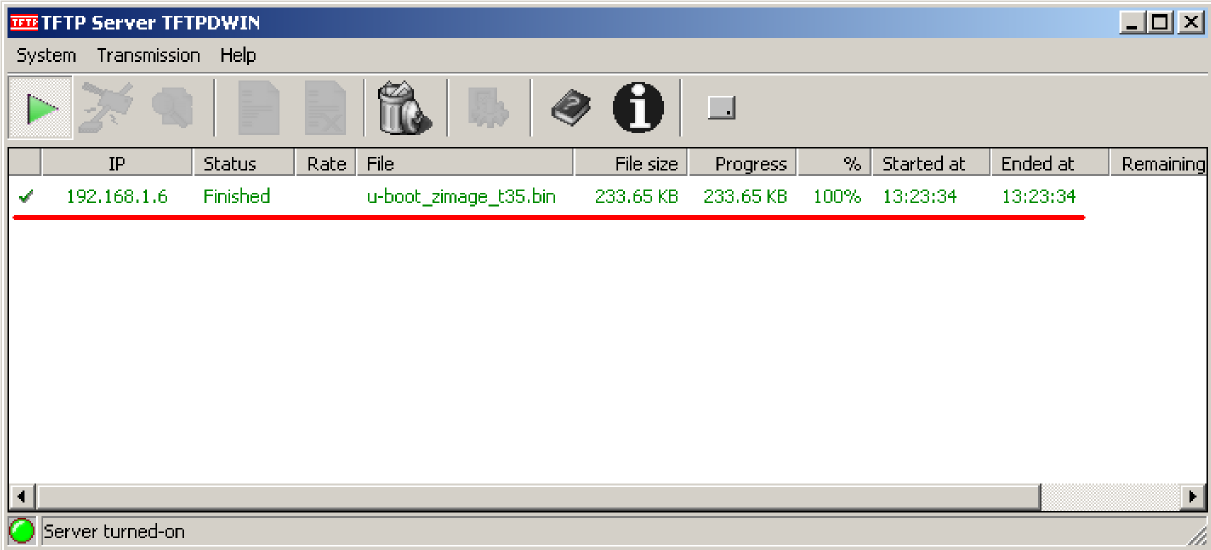
Step13, execute TFTP command to download u-boot to Nand Flash:

The TFTP command: tftp 0x30000000 u-boot_zImage_T35.bin; nand erase bios; nand write.jffs2 0x30000000 bios $(filesize)

Including 3 parts: 1, TFTP downloads "u-boot_zImage_T35.bin" to SDRAM starting from the address 0x30000000; 2, wipe up BIOS partition (from 0x0 to 0x40000) of Nand Flash; 3, save the data begin from the address 0x30000000 to BIOS partition in Nand Flash.



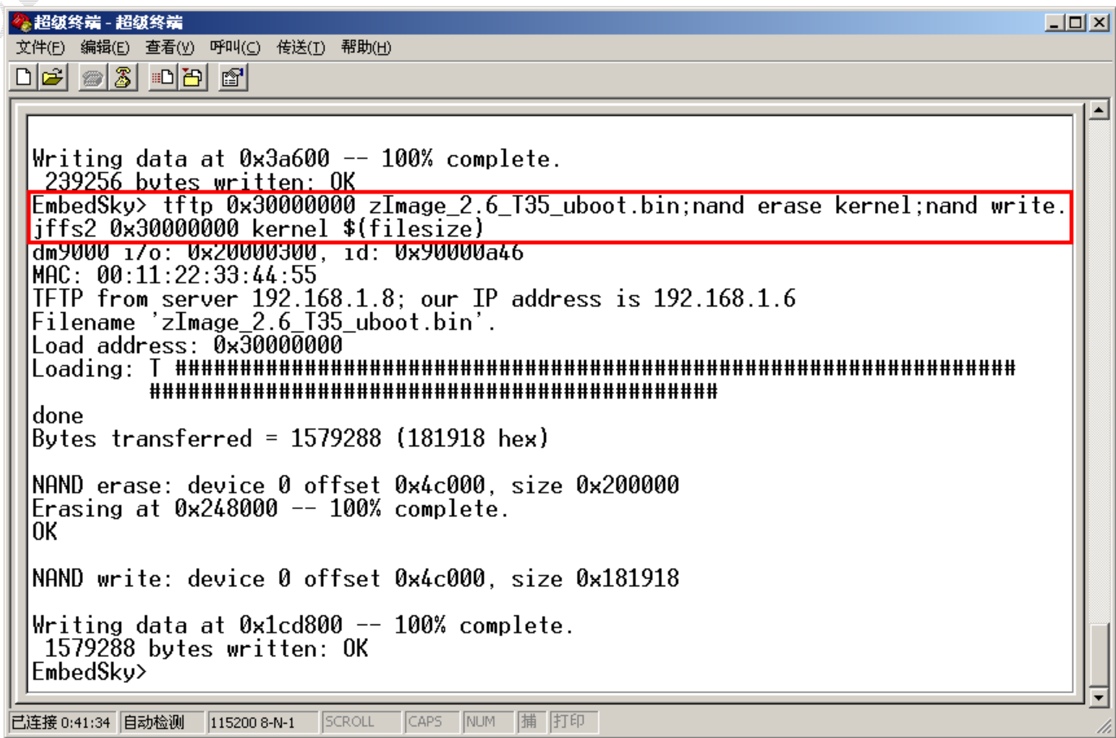
TFTP proxy state is shown in the following diagram:



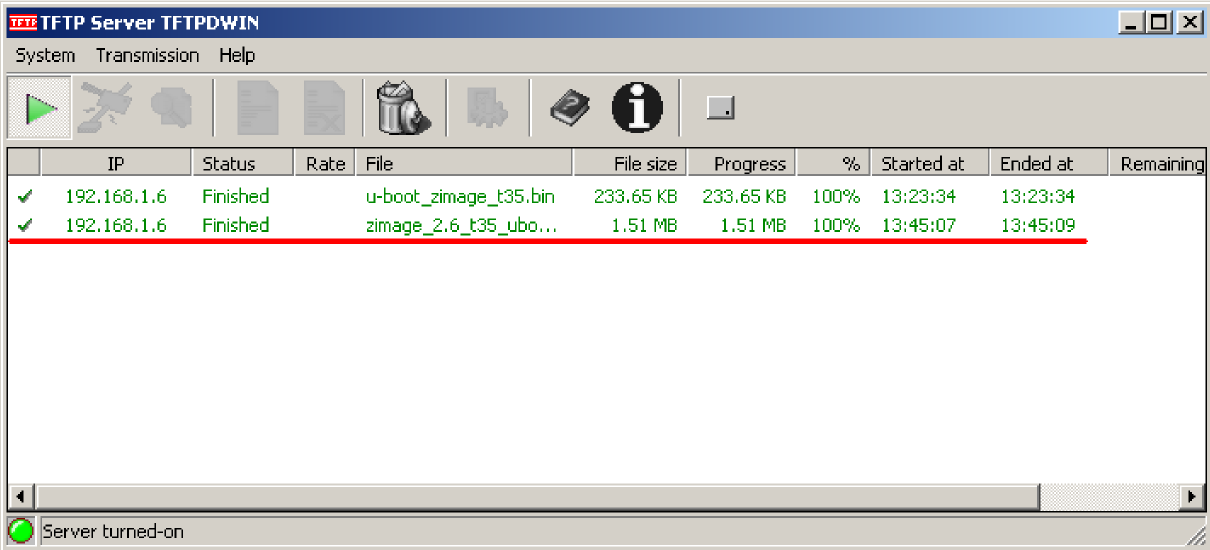
Step14, execute TFTP command to download Linux kernel to Nand Flash:

The TFTP command: tftp 0x30000000 zImage_2.6_T35_uboot.bin; nand erase kernel; nand write.jffs2 0x30000000 kernel $(filesize)

The Linux kernel partition: From 0x4C000 to 0x24C000



TFTP proxy state is shown in the following diagram:

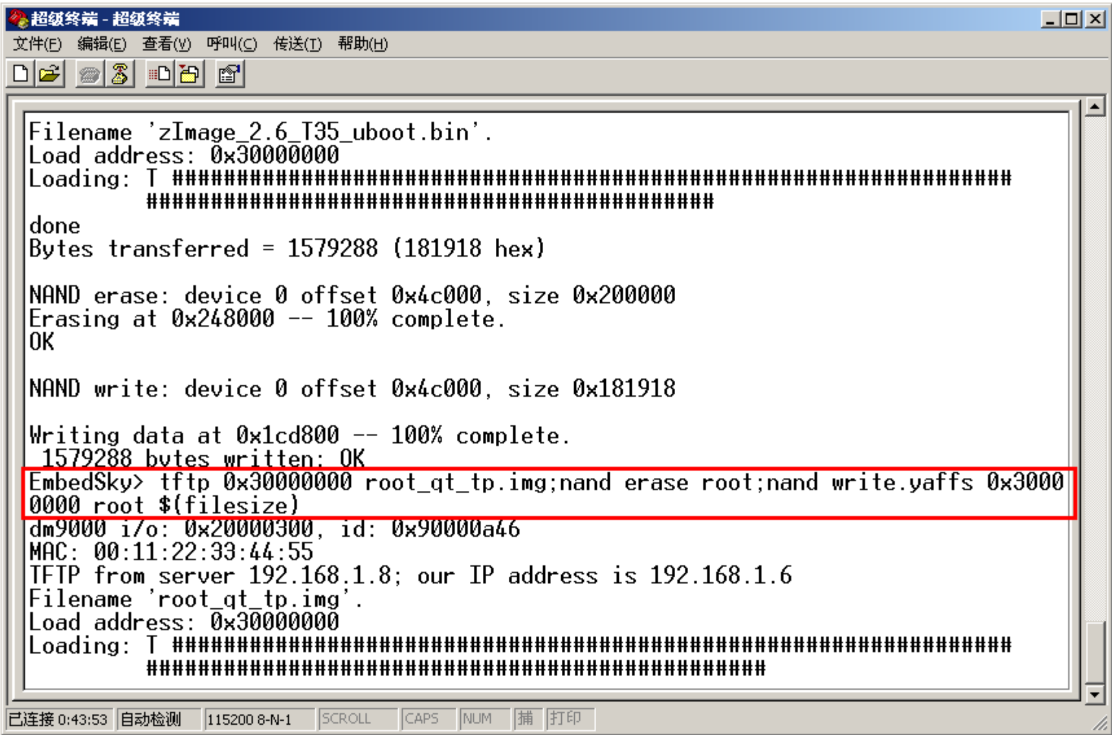


Step15, execute TFTP command to download file system to Nand Flash:

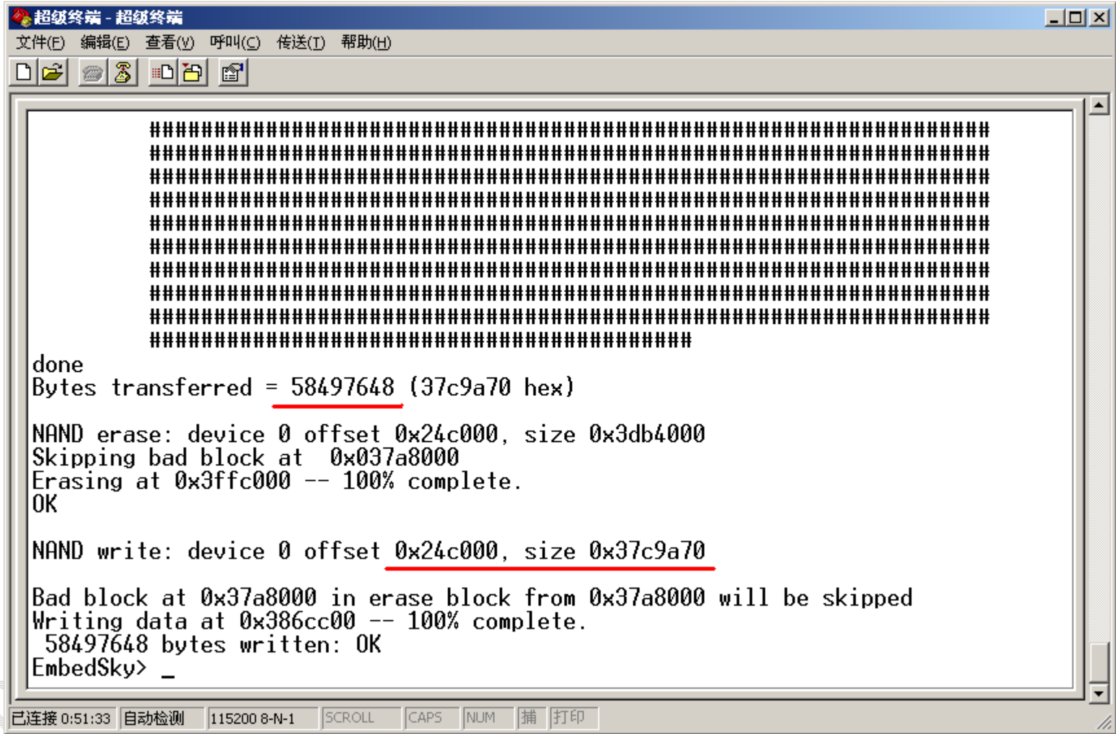
The TFTP command: tftp 0x30000000 root_qt_tp.img; nand erase root; nand write.yaffs 0x30000000 root $(filesize)

The root partition: From 0x24C000 to 0x4000000

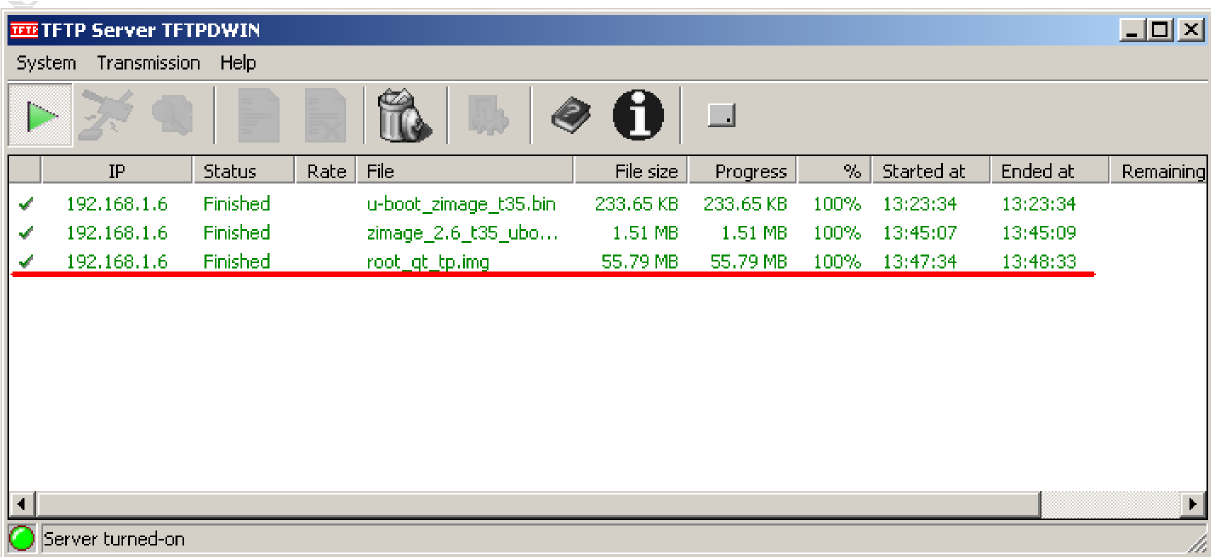
Caution: We use the parameter yaffs here because we are downloading yaffs file system.



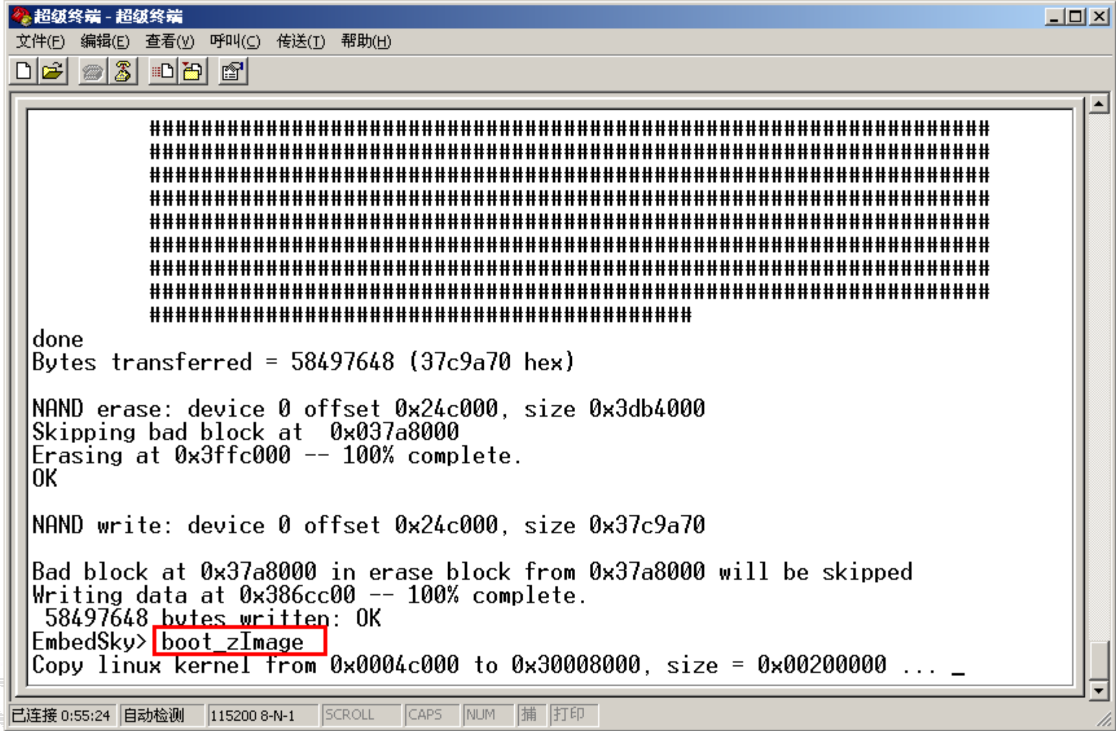
The following diagram indicates that download is complete:



TFTP proxy state is shown in the following diagram:



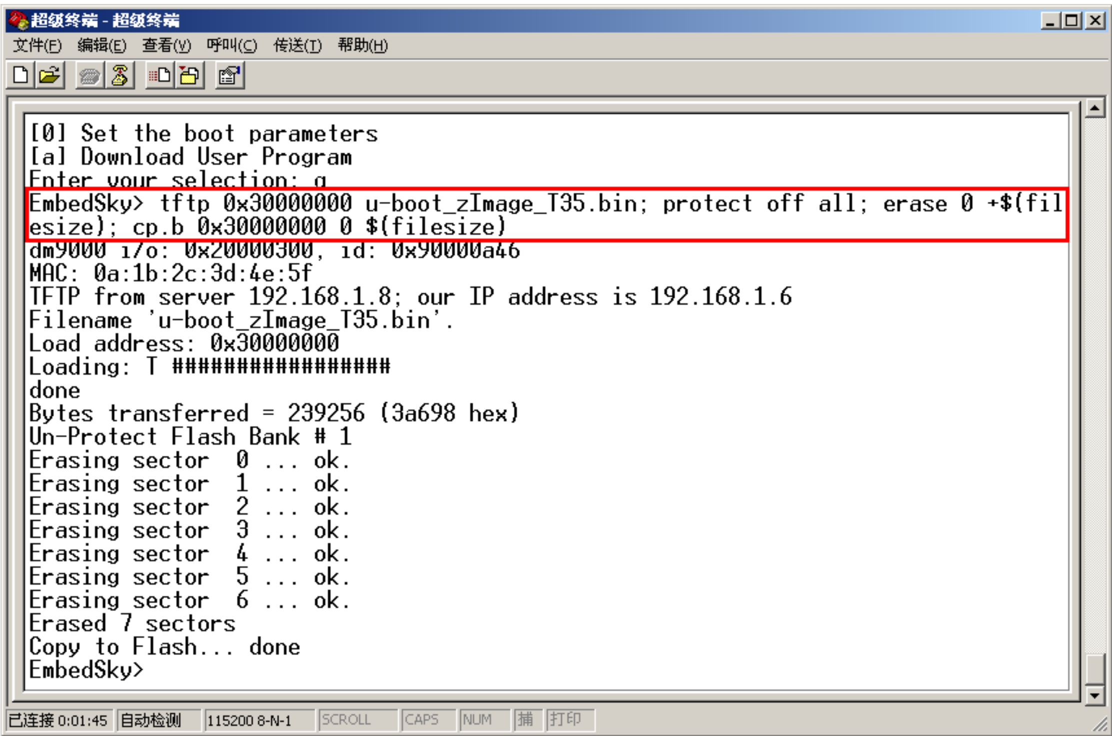
Step16, execute TFTP command "boot_zImage" to start up Linux:



```
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
################################################################
##########################################
done
Bytes transferred = 58497648 (37c9a70 hex)

NAND erase: device 0 offset 0x24c000, size 0x3db4000
Skipping bad block at  0x037a8000
Erasing at 0x3ffc000 -- 100% complete.
OK

NAND write: device 0 offset 0x24c000, size 0x37c9a70

Bad block at 0x37a8000 in erase block from 0x37a8000 will be skipped
Writing data at 0x386cc00 -- 100% complete.
 58497648 bytes written: OK
EmbedSky> boot_zImage
Copy linux kernel from 0x0004c000 to 0x30008000, size = 0x00200000 ... _
```

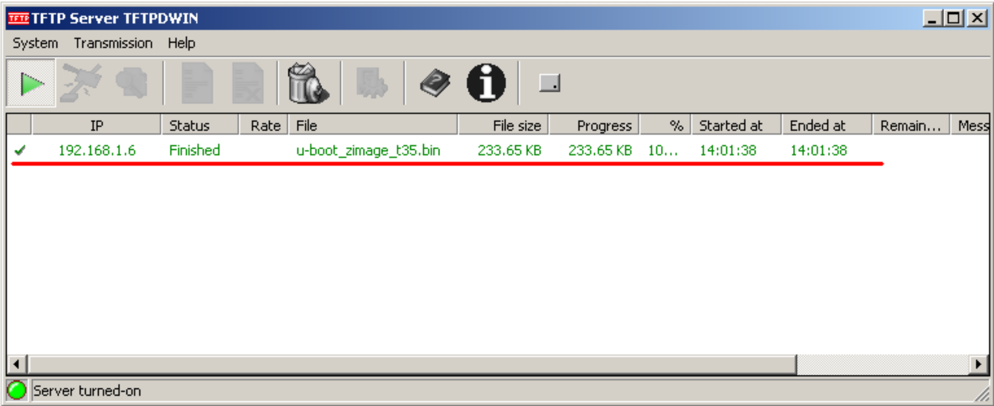
Step17, execute TFTP command to download u-boot to Nor Flash (start-up from Nor Flash is the prerequisite):

The TFTP command: tftp 0x30000000 u-boot_zImage_T35.bin; protect off all; erase 0 +$(filesize); cp.b 0x30000000 0 $(filesize). Be cautious of the change here.



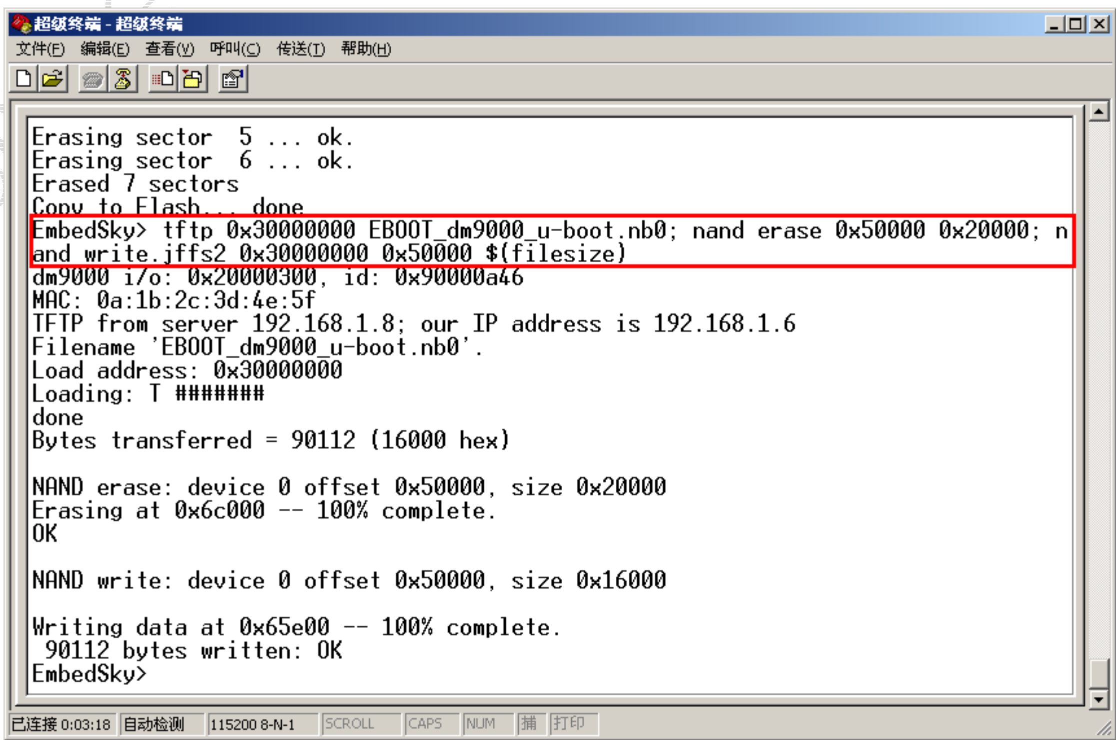
```
[0] Set the boot parameters
[a] Download User Program
Enter your selection: q
EmbedSky> tftp 0x30000000 u-boot_zImage_T35.bin; protect off all; erase 0 +$(fil
esize); cp.b 0x30000000 0 $(filesize)
dm9000 i/o: 0x20000300, id: 0x90000a46
MAC: 0a:1b:2c:3d:4e:5f
TFTP from server 192.168.1.8; our IP address is 192.168.1.6
Filename 'u-boot_zImage_T35.bin'.
Load address: 0x30000000
Loading: T #################
done
Bytes transferred = 239256 (3a698 hex)
Un-Protect Flash Bank # 1
Erasing sector  0 ... ok.
Erasing sector  1 ... ok.
Erasing sector  2 ... ok.
Erasing sector  3 ... ok.
Erasing sector  4 ... ok.
Erasing sector  5 ... ok.
Erasing sector  6 ... ok.
Erased 7 sectors
Copy to Flash... done
EmbedSky>
```

TFTP proxy state is shown in the following diagram:

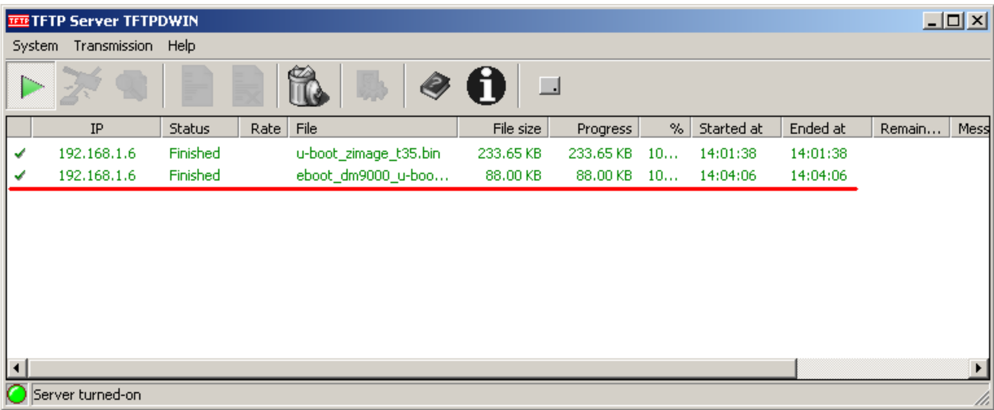


Step18, execute TFTP command to download eboot to Nand Flash:

The TFTP command: tftp 0x30000000 EBOOT_dm9000_u-boot.nb0; nand erase 0x50000 0x20000; nand write.jffs2 0x30000000 0x50000 $(filesize). 0x50000 is eboot start address in Nand Flash, and 0x20000 is the length of eboot partition.

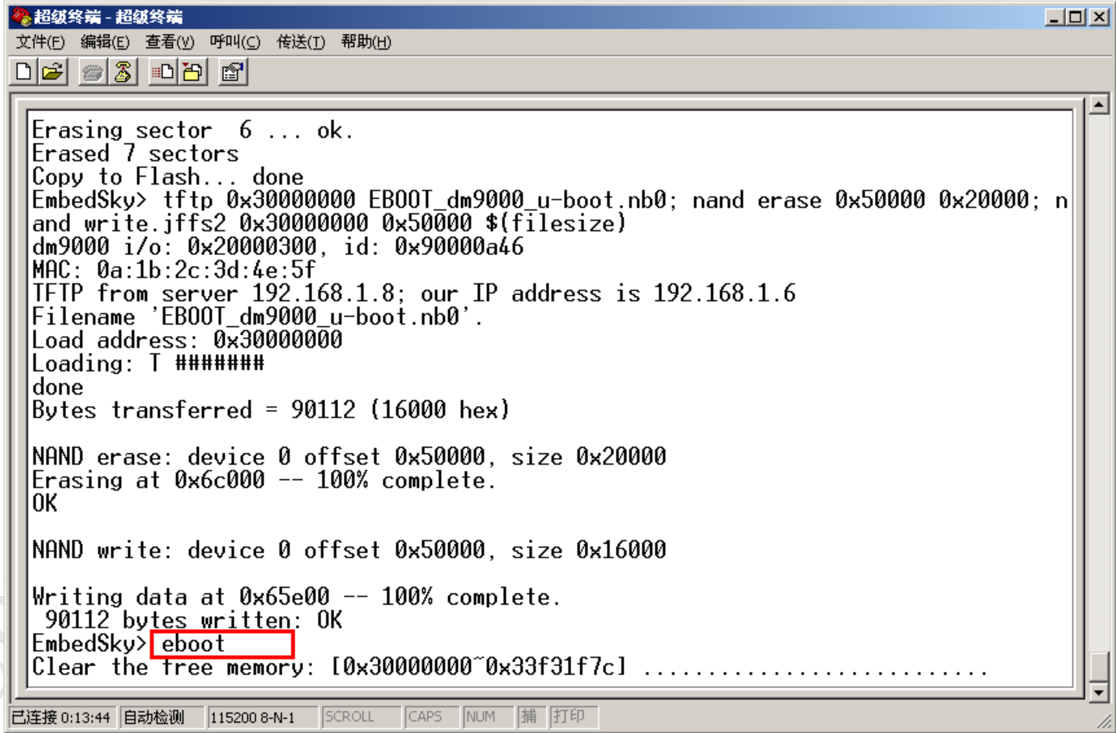


TFTP proxy state is shown in the following diagram:

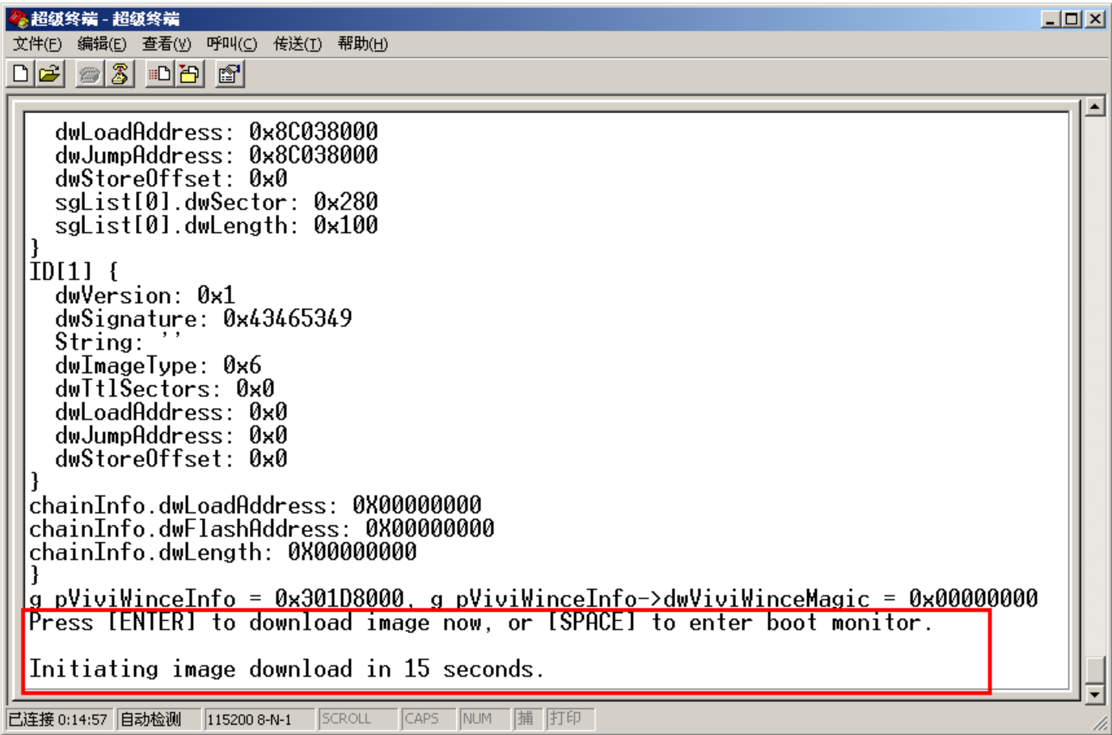


Step19, execute TFTP function of eboot to download and burn WinCE image:

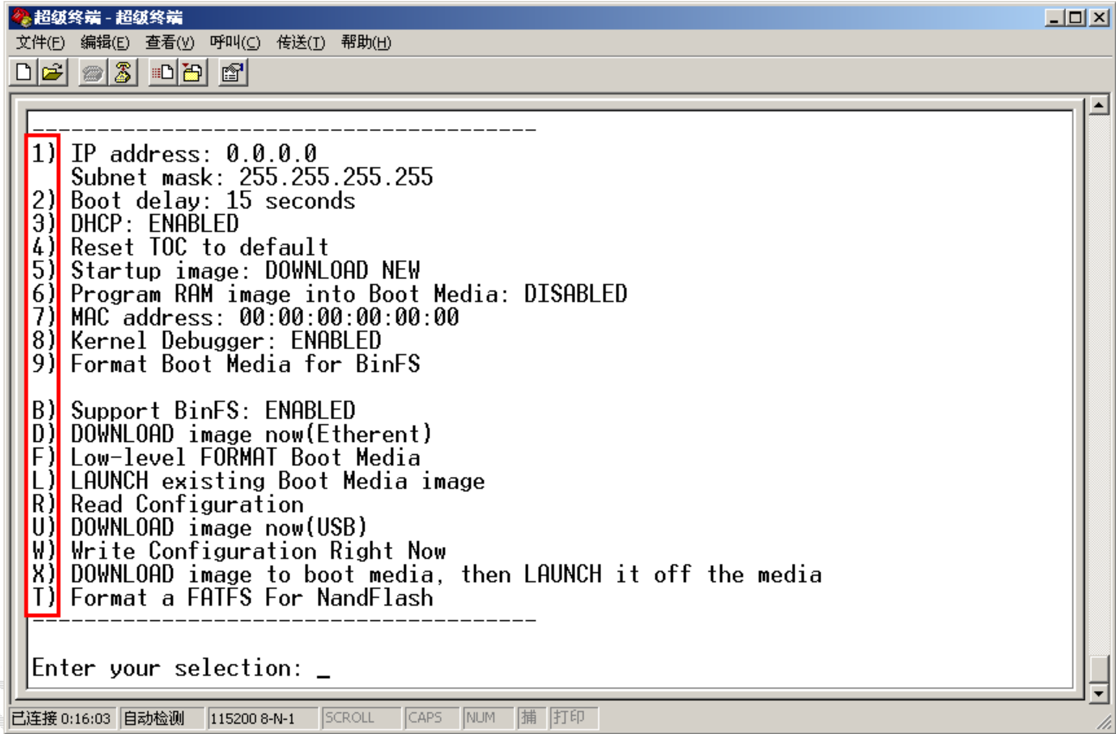
Start up eboot in u-boot operation interface. The command is: eboot.



The following diagram display eboot start-up process:

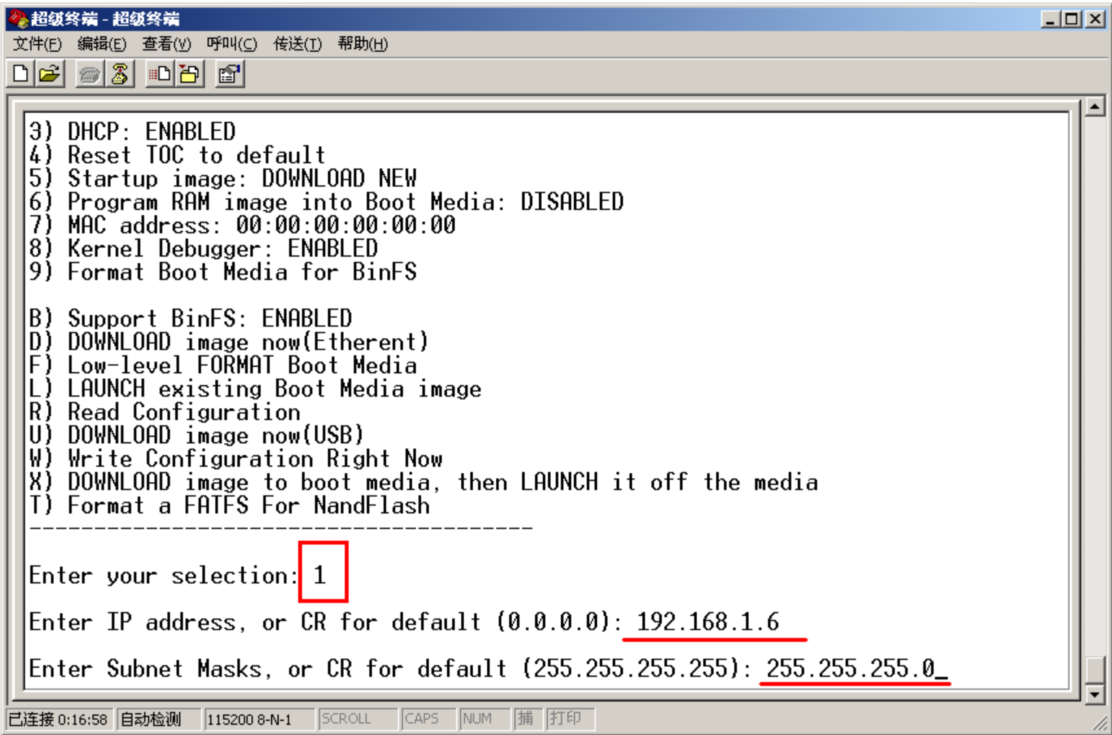


Enter space-key to enter into eboot download mode:



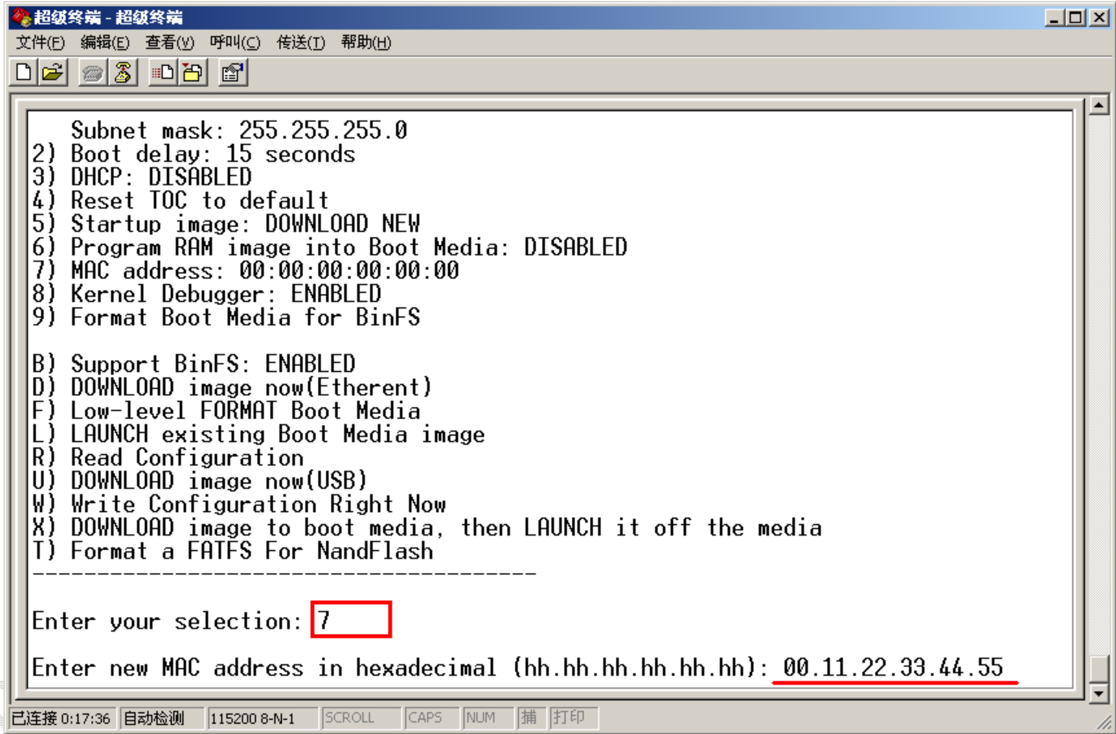
```
超级终端 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
---------------------------------------
1) IP address: 0.0.0.0
   Subnet mask: 255.255.255.255
2) Boot delay: 15 seconds
3) DHCP: ENABLED
4) Reset TOC to default
5) Startup image: DOWNLOAD NEW
6) Program RAM image into Boot Media: DISABLED
7) MAC address: 00:00:00:00:00:00
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS

B) Support BinFS: ENABLED
D) DOWNLOAD image now(Etherent)
F) Low-level FORMAT Boot Media
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
X) DOWNLOAD image to boot media, then LAUNCH it off the media
T) Format a FATFS For NandFlash
---------------------------------------

Enter your selection: _
已连接 0:16:03 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

Enter “1” to set IP address and Subnet Masks of platform:

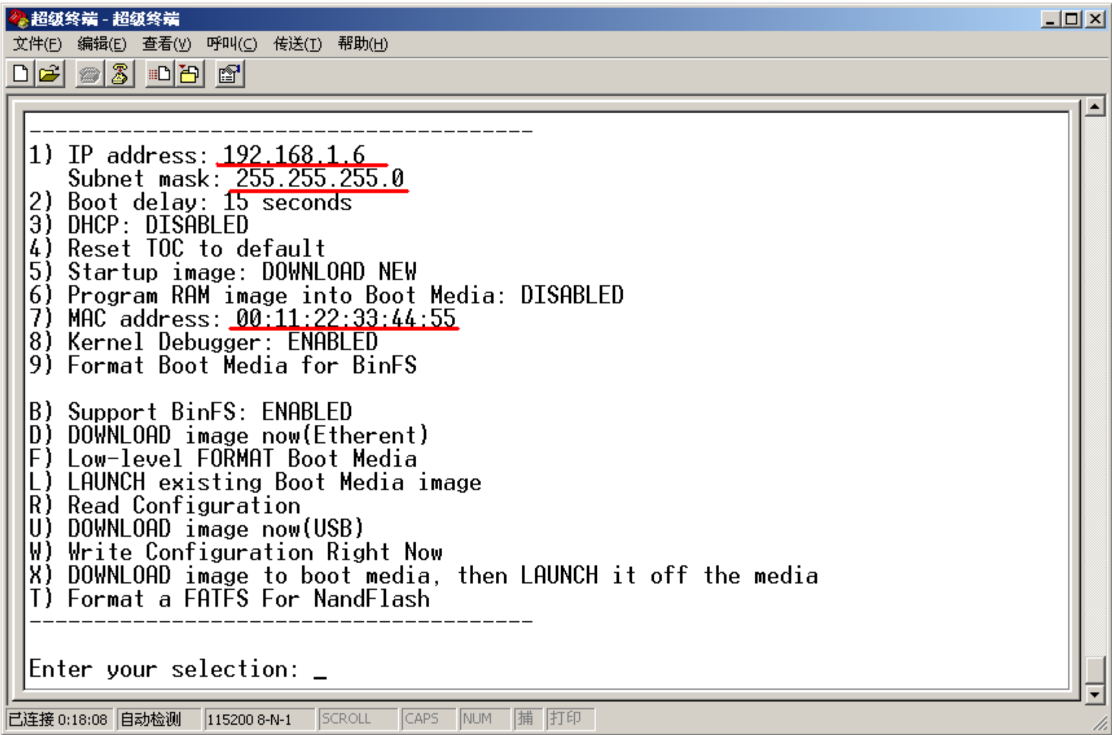


```
超级终端 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
3) DHCP: ENABLED
4) Reset TOC to default
5) Startup image: DOWNLOAD NEW
6) Program RAM image into Boot Media: DISABLED
7) MAC address: 00:00:00:00:00:00
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS

B) Support BinFS: ENABLED
D) DOWNLOAD image now(Etherent)
F) Low-level FORMAT Boot Media
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
X) DOWNLOAD image to boot media, then LAUNCH it off the media
T) Format a FATFS For NandFlash
---------------------------------------

Enter your selection: 1

Enter IP address, or CR for default (0.0.0.0): 192.168.1.6

Enter Subnet Masks, or CR for default (255.255.255.255): 255.255.255.0_
已连接 0:16:58 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

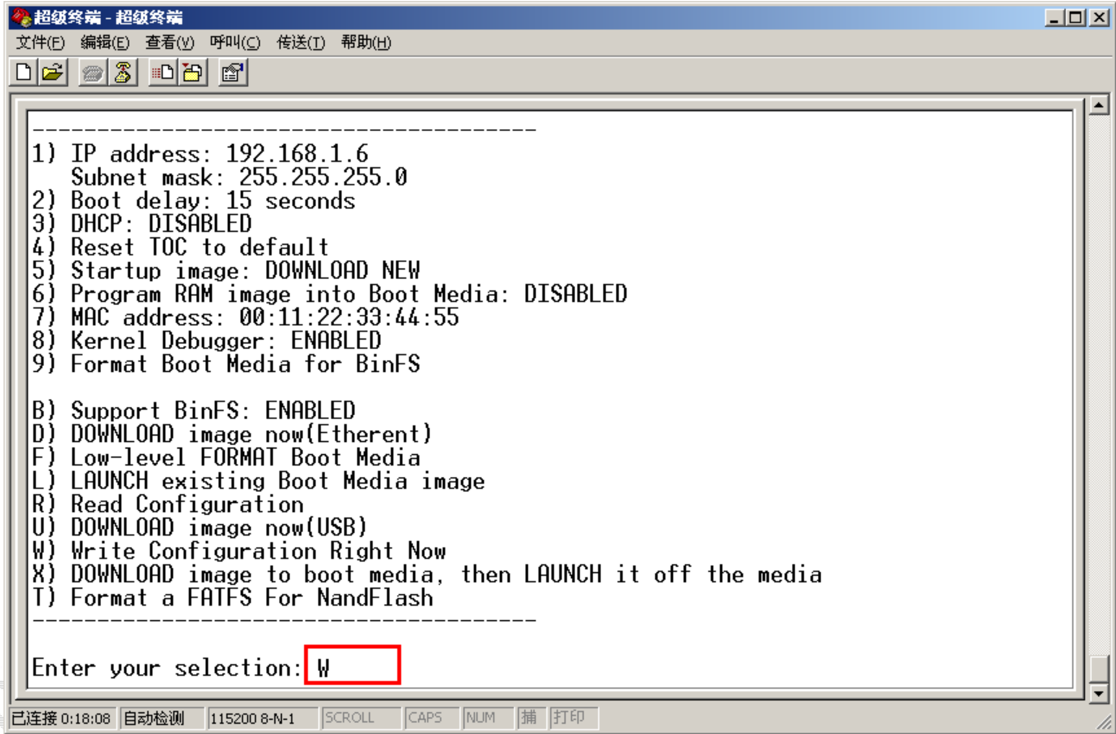
Enter “7” to set MAC address:



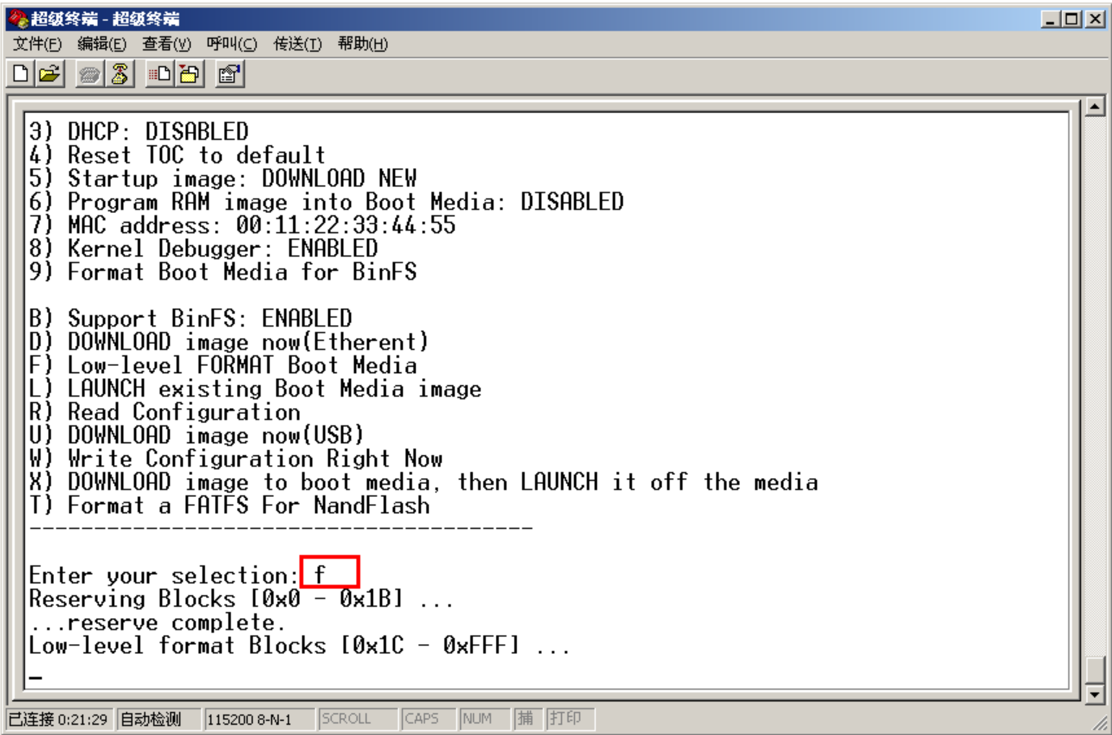
The following diagram indicates that the setting is complete:



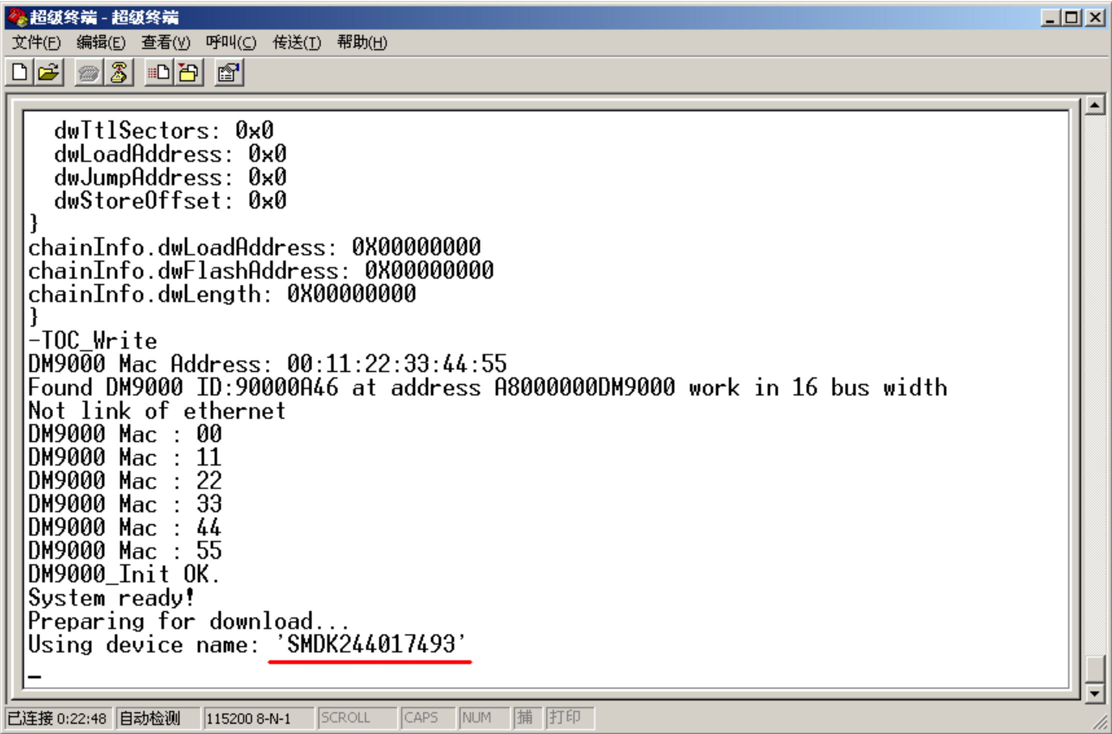
Enter “W” to save all the settings:



Enter “F” to low-level format Nand Flash (no case sensitive):

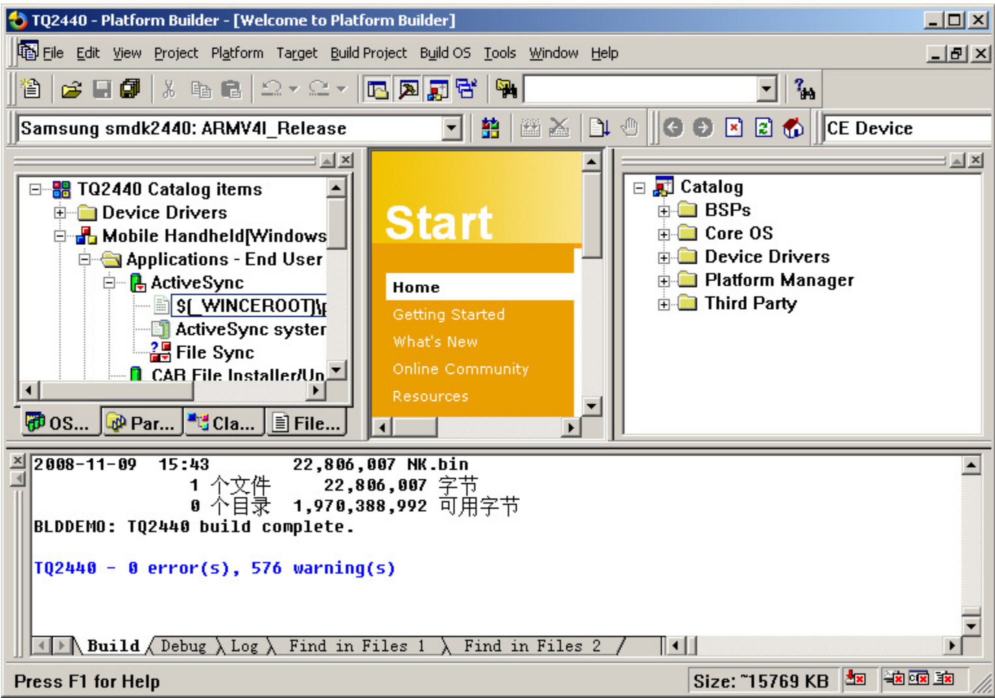


Enter “X” (no case sensitive) to download “NK.bin” to SDRAM and then burn it to Nand Flash, as shown in the following diagram:

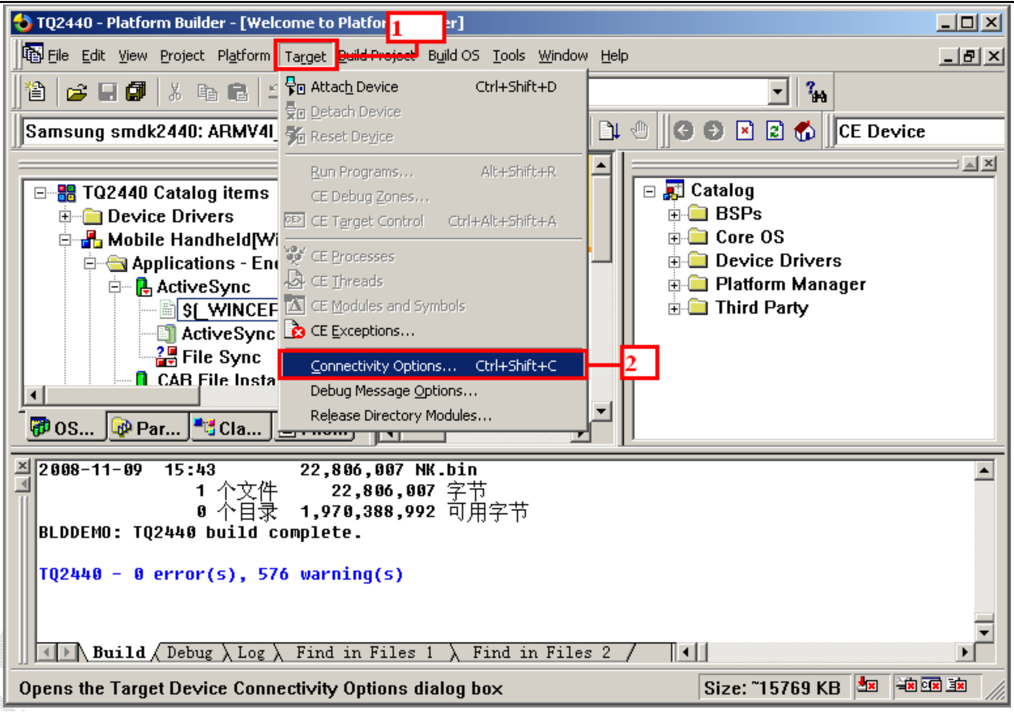


The phrase “SMDK244017493” in the upper diagram represents the device in PB.

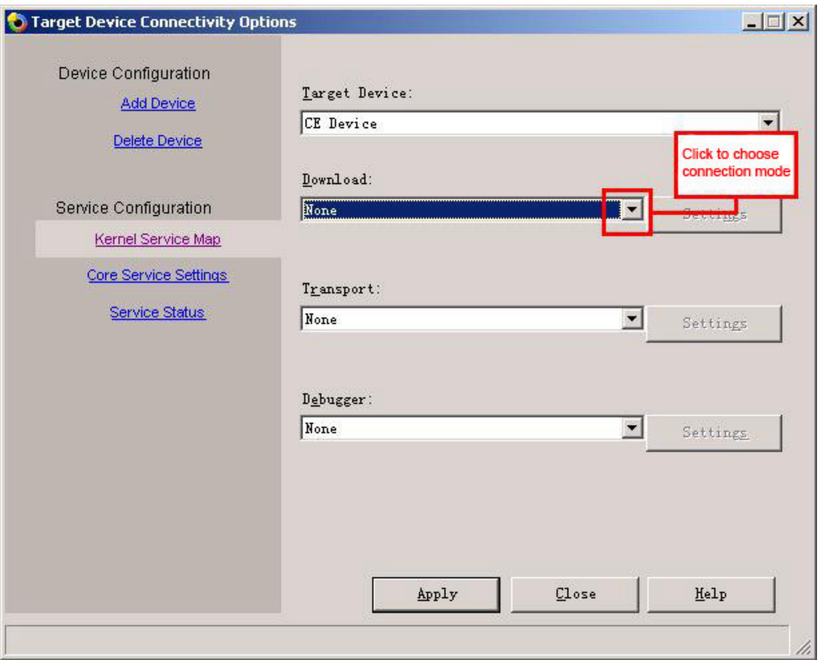
Start up “Platform Builder 5.0”, and load project files (the project files loaded here needs to be compiled):



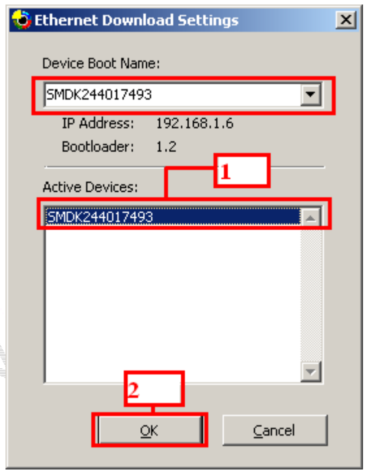
Click “Target->Connectivity Options” to configure PB of TFTP, as shown in the following diagram:



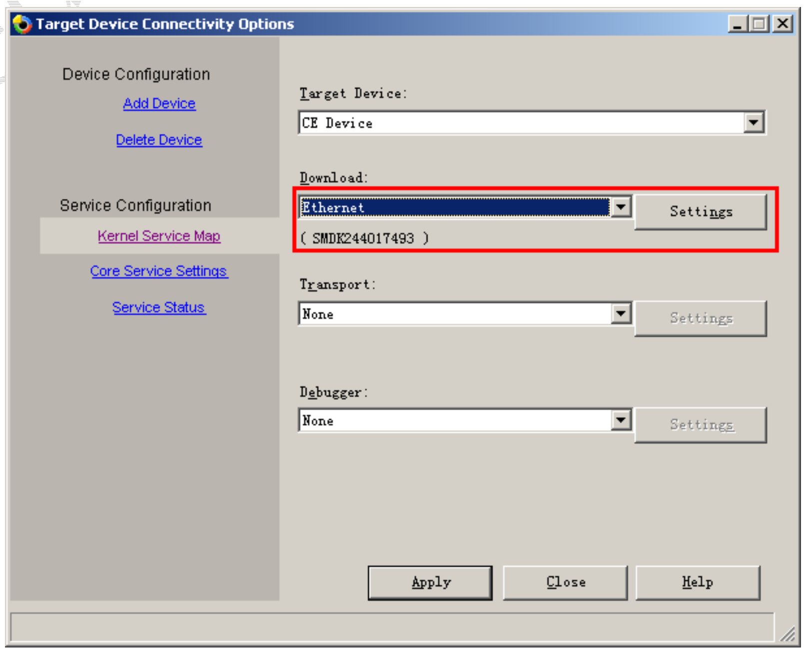
Select “Ethernet” under “Download”:



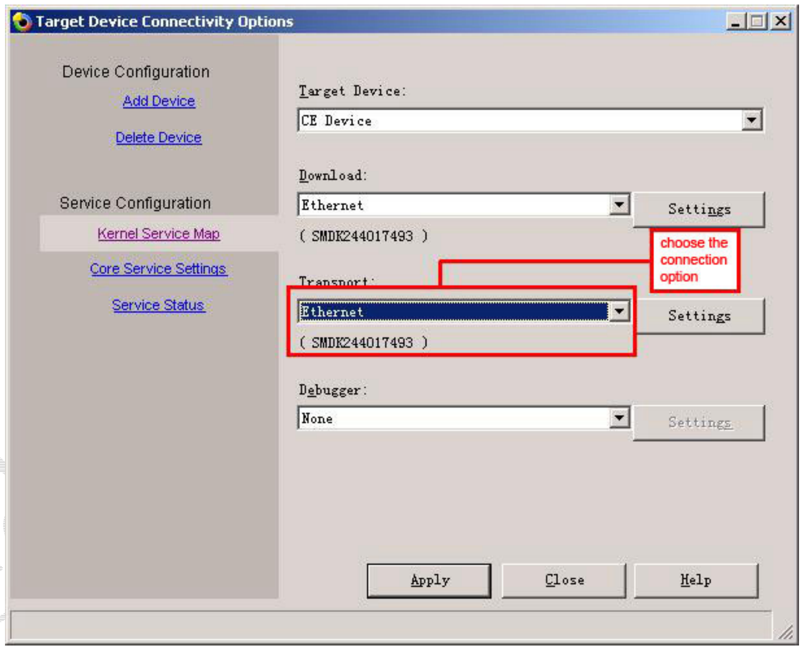
Click “Settings”. Enter the phrase “SMDK244017493” gotten previously and click “Ok”:



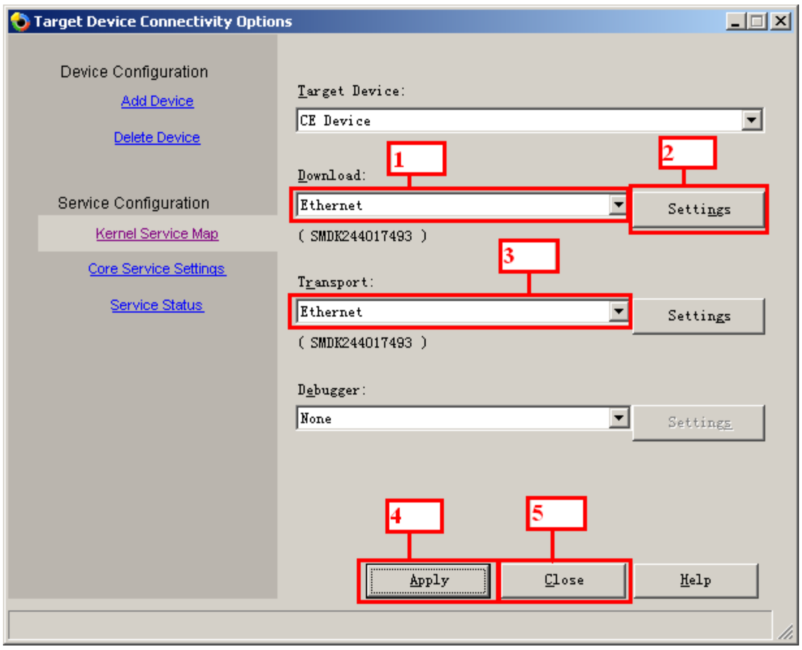
Download mode configuration is complete, as shown in the following diagram:



Select “Ethernet” under “Transport”, as the following diagram:

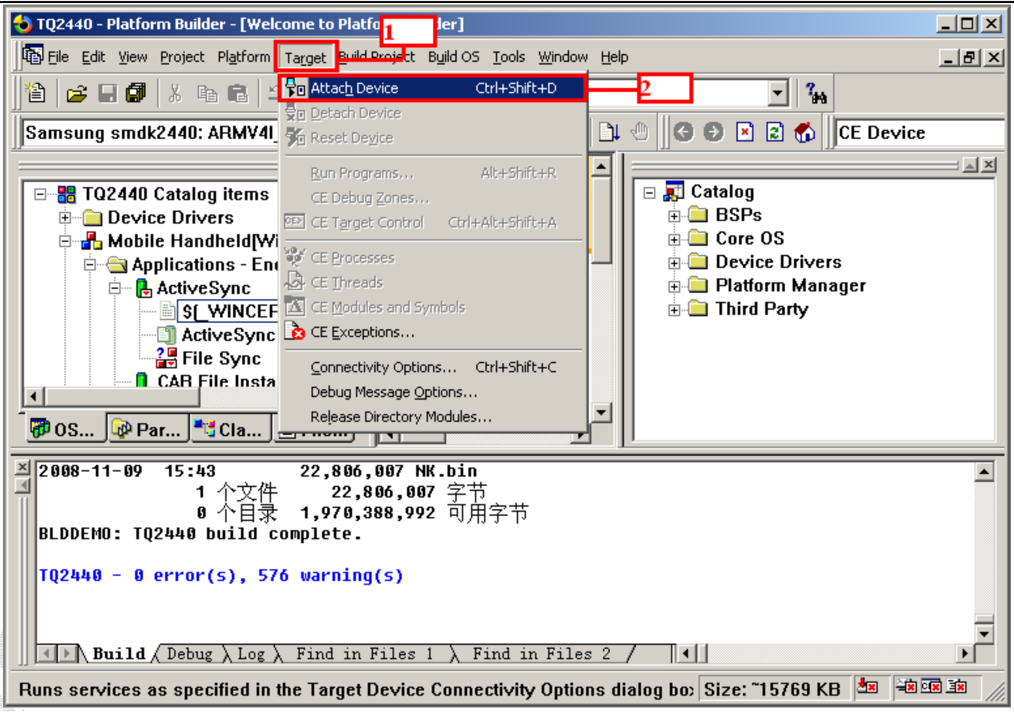


The configuration interface appears as the following diagram:

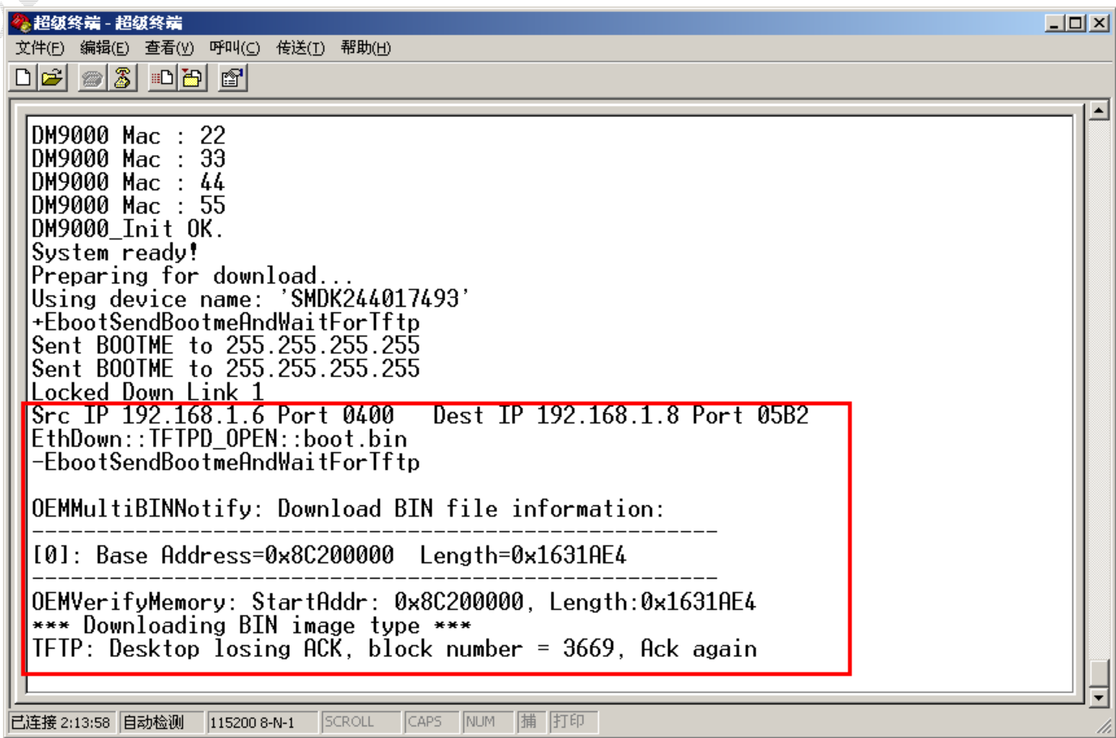


Click “Apply” and click “Close” to save and exit.

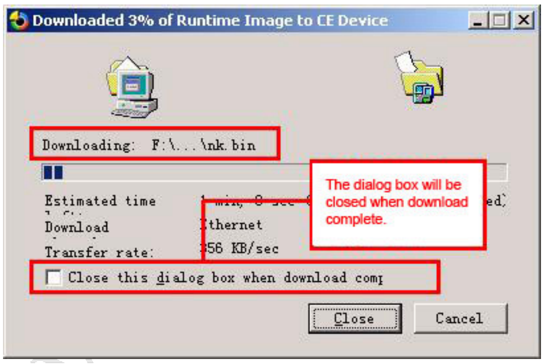
Click “Target->Attach Device” to start TFTP transmission:



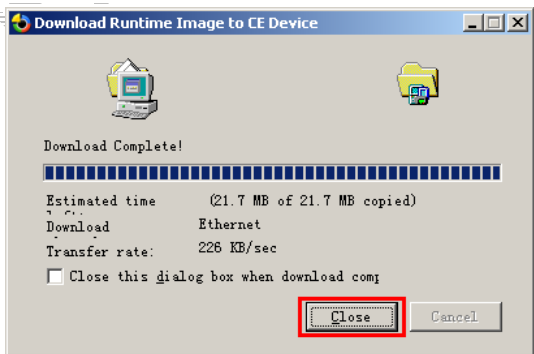
PB starts up TFTP to commucate with eboot after about 1 second, as shown in the following diagram:



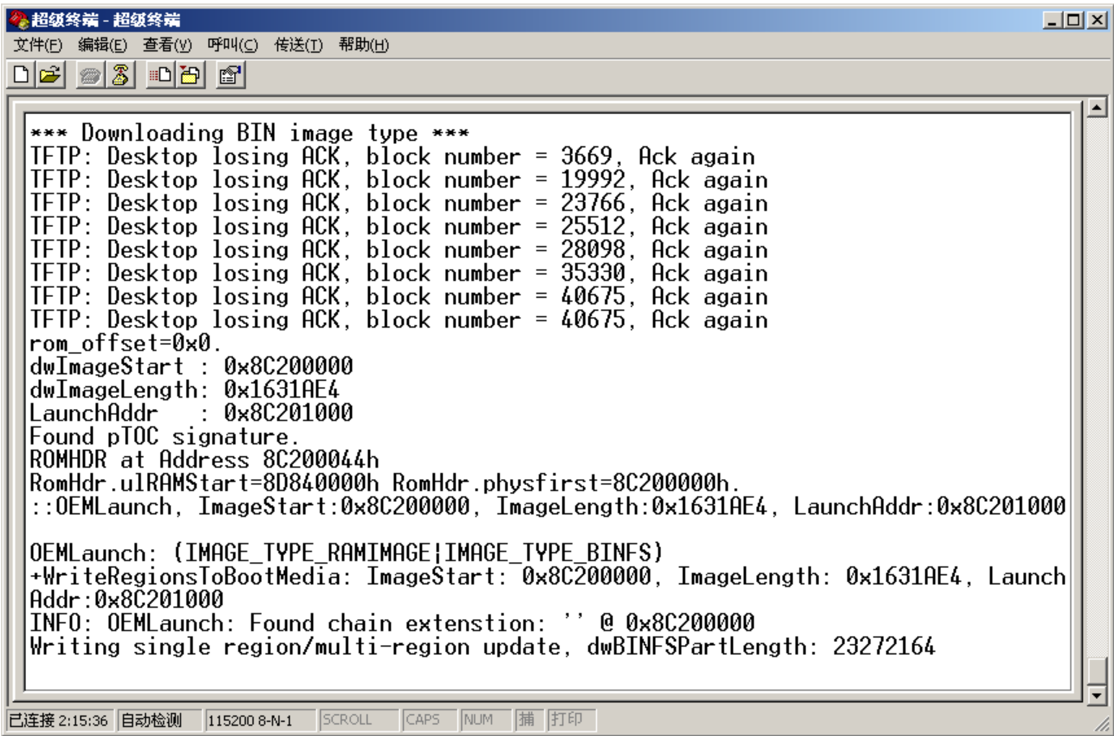
The transmission interface appears, as shown in the following diagram:



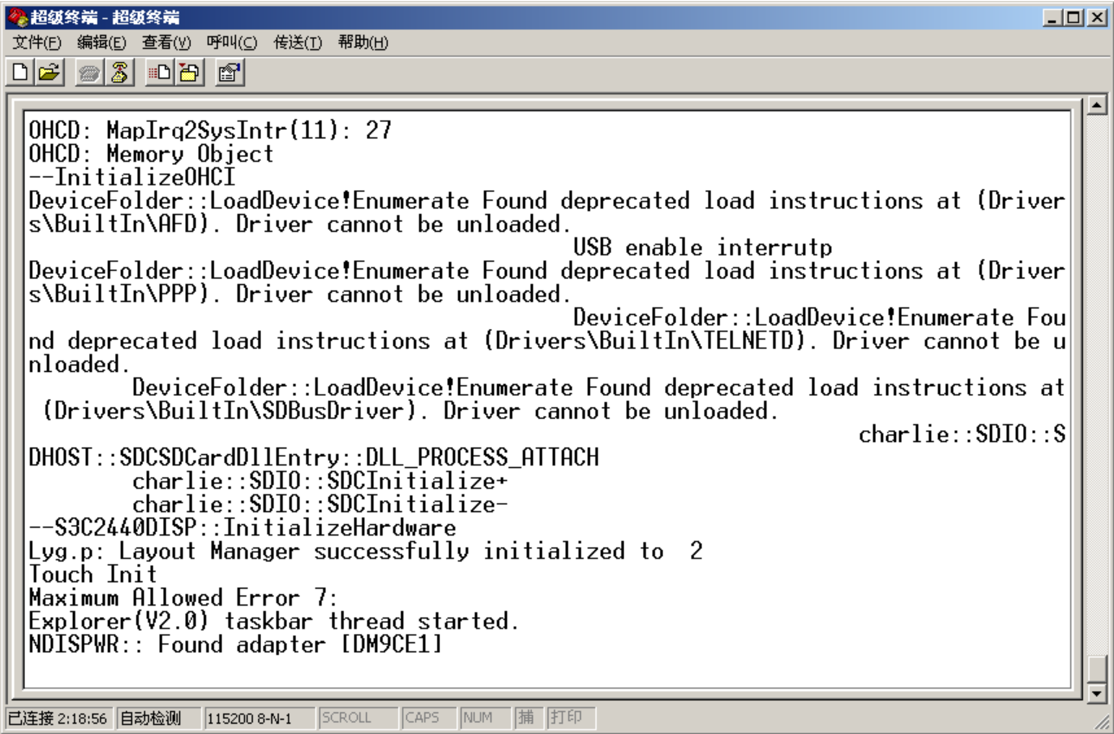
Click “Close” to exit after stransmission is complete:



eboot begins burning and boot WinCE automatically. This process needs about 4 minutes, as shown in the following diagram:



The following diagram displays the serial port information when WinCE starts boot:



The introduction of u-boot is finished here. You can consult www.embedsky.net/bbs if you have any more questions

3. 5 Burning system

Caution: Make sure that u-boot has been burned to Flash and works well before you try to burn operating system. If there is no u-boot in Flash, please consult “3.2 节” and “3.3 节” to burn u-boot first.

u-boot has been burned to Flash in factory, so there is no need to burn u-boot again.

3. 5. 1 Burning Linux OS

Please consult Step 3, Step 5 or Step 6 of “3.4.3 节” when you try to burn Linux OS.

3. 5. 2 Burning WinCE OS

Please consult Step 2 or Step 4 of “3.4.3 节” when you try to burn WinCE OS.

3. 5. 3 Burning TQ2440-Test_xxxx.bin file

Please consult “3.2 节” when burning. The start address is 0 at Nor Flash.

If you want to burn “TQ2440-Test_xxxx.bin” to Nand Flash, you could choose to enter “a” in u-boot one-key menu, or execute the TFTP command of u-boot. The command is: tftp 0x30000000 TQ2440_Test_xxxx.bin; nand erase 0x0 0x92000; nand write.jffs2 0x30000000 0x0 $(filesize)

Caution: the value “0x92000” needs to be modified according to the actual size of “TQ2440-Test_xxxx.bin”

3. 6 Linux experiment

3. 6. 1 Experiment of program termination

2 method to terminate a program:

Method 1: In terminal console, press “Ctrl” and hold, and then press “C”.

Method 2: If the program is running background, you can first execute the command “ps” to search the process ID of the program, and then execute the command “kill” to terminal it.

3. 6. 2 Experiment of program auto-run configuration

You can configure the booting script or other system settings to set auto-run. The booting script is under the directory “/etc/init.d/rcS”, the settings is shown as the following contents (the following contents might have some tiny differences with the actual ones):

```
#! /bin/sh

PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:        #Set the default execution path
runlevel=S                                                #User level. Here is: Single
prevlevel=N
umask 022
export PATH runlevel prevlevel

#
#     Trap CTRL-C &c only in this shell so we can interrupt subprocesses.
#
trap ":" INT QUIT TSTP
hwclock –s                                                #Synchronize Linux clock and RTC

mknod /dev/pts/0 c 136 0
ln -s /dev/v4l/video0 /dev/video0
ln -s /dev/fb/0 /dev/fb0                                  #The symbol link of FrameBuffer
ln -s /dev/vc/0 /dev/tty1
ln -s /dev/sound/dsp /dev/dsp                             #The symbol link of sound device
ln -s /dev/sound/mixer /dev/mixer
ln -s /dev/input/tsraw0 /dev/h3600_tsraw

#Set the common temporary directory
mount -t proc none /proc
```

```
mount -t tmpfs none /tmp
mount -t tmpfs none /var

mkdir -p /var/lib
mkdir -p /var/run
mkdir -p /var/log

/etc/rc.d/init.d/netd start                 #Start telnet/ftp service
/etc/rc.d/init.d/httpd start                #Start web server
/etc/rc.d/init.d/leds start                 #Start LED

ifconfig lo 127.0.0.1                       #IP address of local loop device
route add default gw 192.168.1.2            #Set gateway
ifconfig eth0 192.168.1.6 up                #Set and enable the IP address of network card

/bin/qtopia &                               #Run Qt/Embeded after start-up

/bin/hostname -F /etc/sysconfig/HOSTNAME
```

3. 6. 3 Experiment of setting and saving system real-time clock

Time in Linux is modified by executing the command "date". The command "hwclock" is used to synchronize S3C2440 internal clock and Linux system clock:

- The format of "date" command: month date hour minute year. For example 2007-08-28 12:30, the command is "date -s 082812302007".
- Execute the command "hwclock -w" to save the time to S3C2440 internal RTC.
- Execute the command "hwclock -s" when Linux start-up to recover the RTC time. You can also add the command to the directory "etc/init.d/rcS" to set auto-run when start-up.

Caution: We have added the command "hwclock -s" to the file "rcS" in factory.

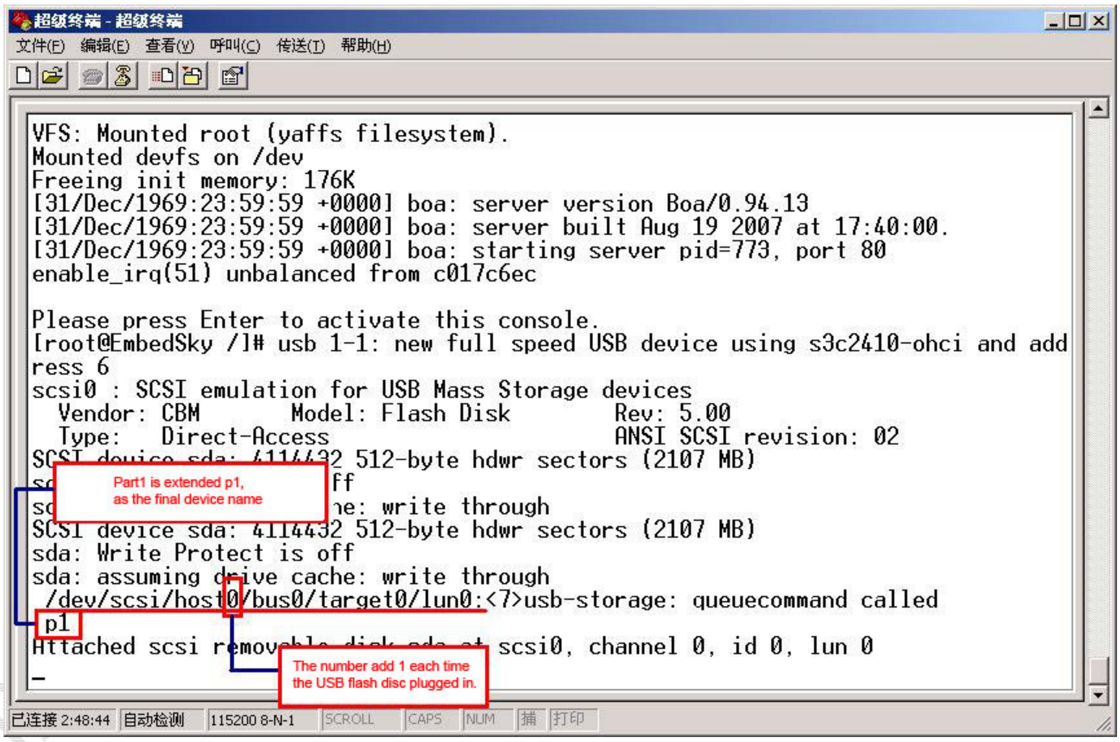
3. 6. 4 Experiment of USB mobile storage device test

The device file corresponding to USB mobile storage device in Linux is "/dev/scsi/host(N-1)/bus0/target0/lun0/part1". Caution: the red N indicates the times you insert USB mobile storage device.

After the USB mobile storage device is inserted, the hyper-terminal appears prompt information, as shown in the following diagram. According to the prompt, you could mount the USB mobile storage device under the directory "/mnt":

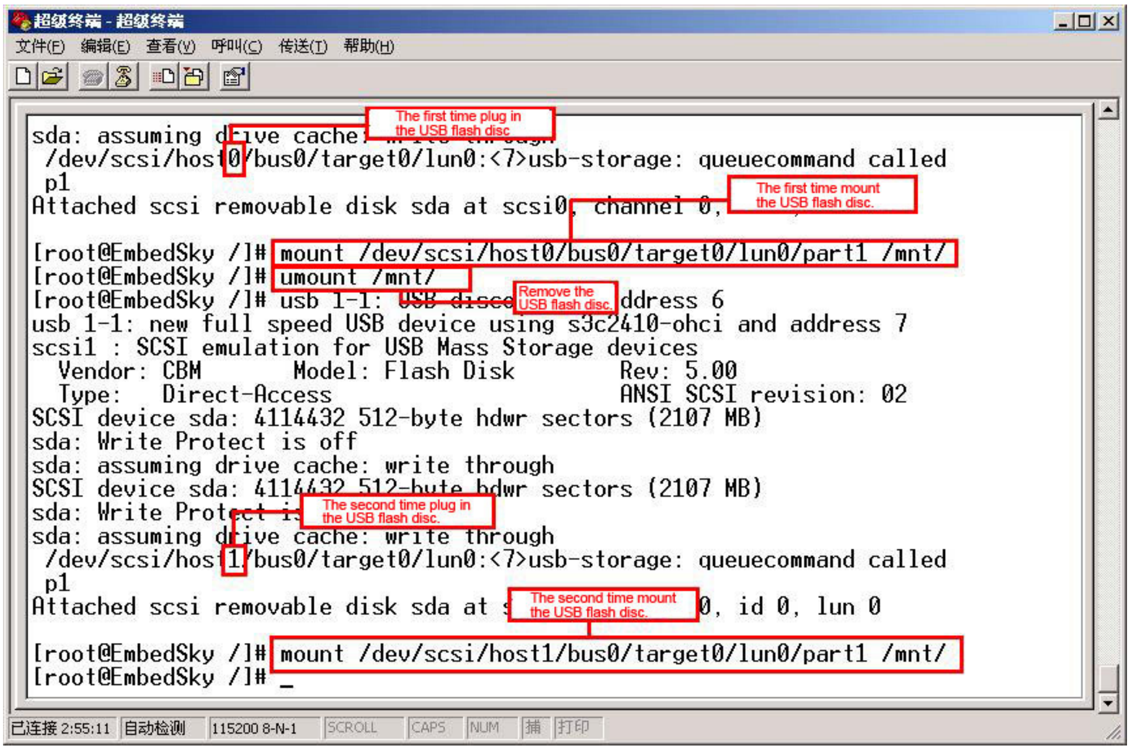
(caution: The number in green frame indicates how many times the USB mobile storage device has been inserted. The number will increase by 1 automatically per insertion)

The prompt of U-disk insertion is shown in the following diagram:



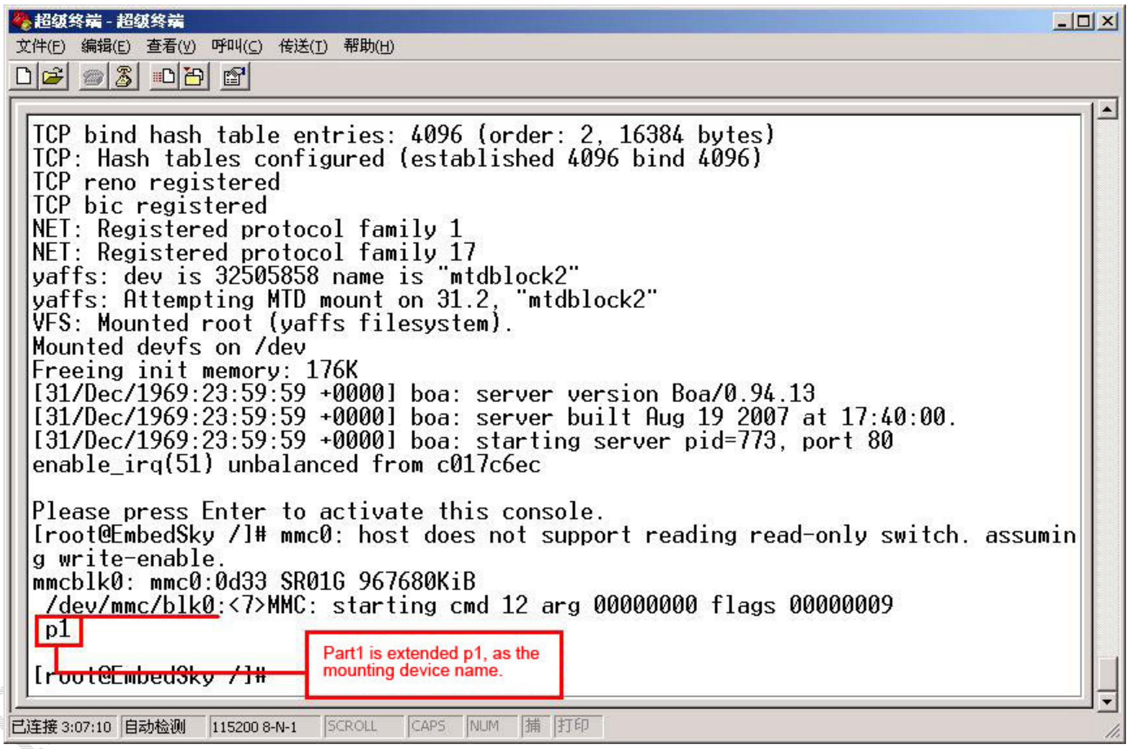
Execute the mount command:

#mount /dev/scsi/host(N-1)/bus0/target0/lun0/part1 /mnt

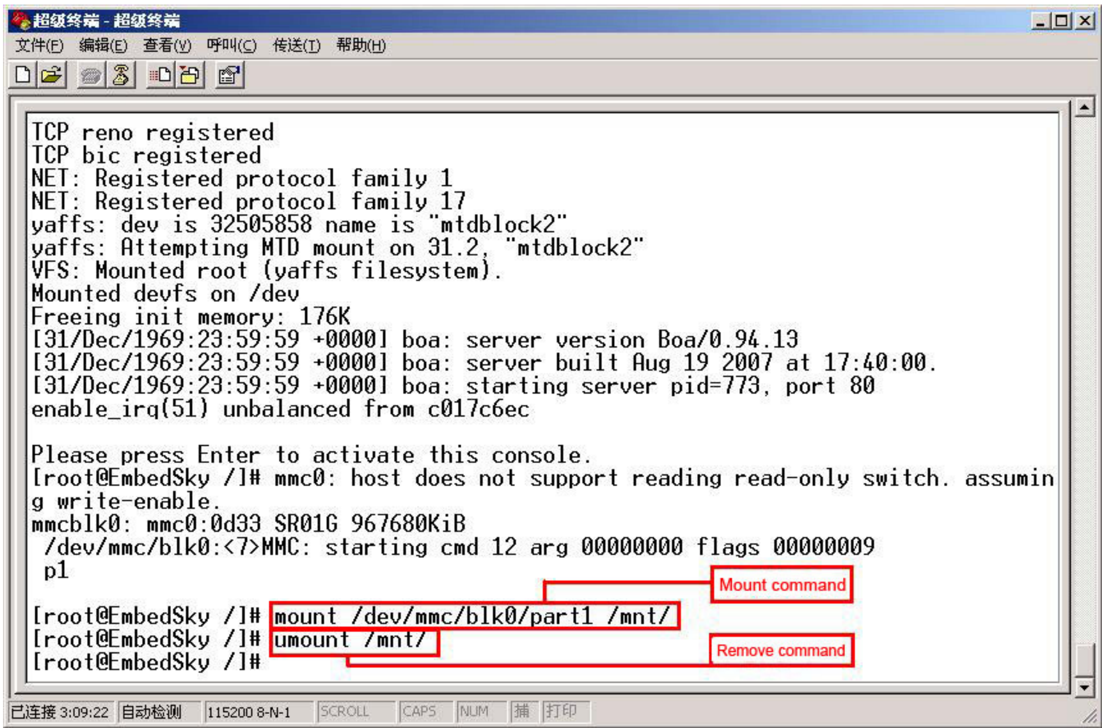


3. 6. 5 Experiment of SD card test

Just like using U-disk, the following information appears after the SD card insertion:



Execute the command “#mount /dev/mmc/blk0/part1 /mnt” to mount SD card under the directory of “/mnt”:

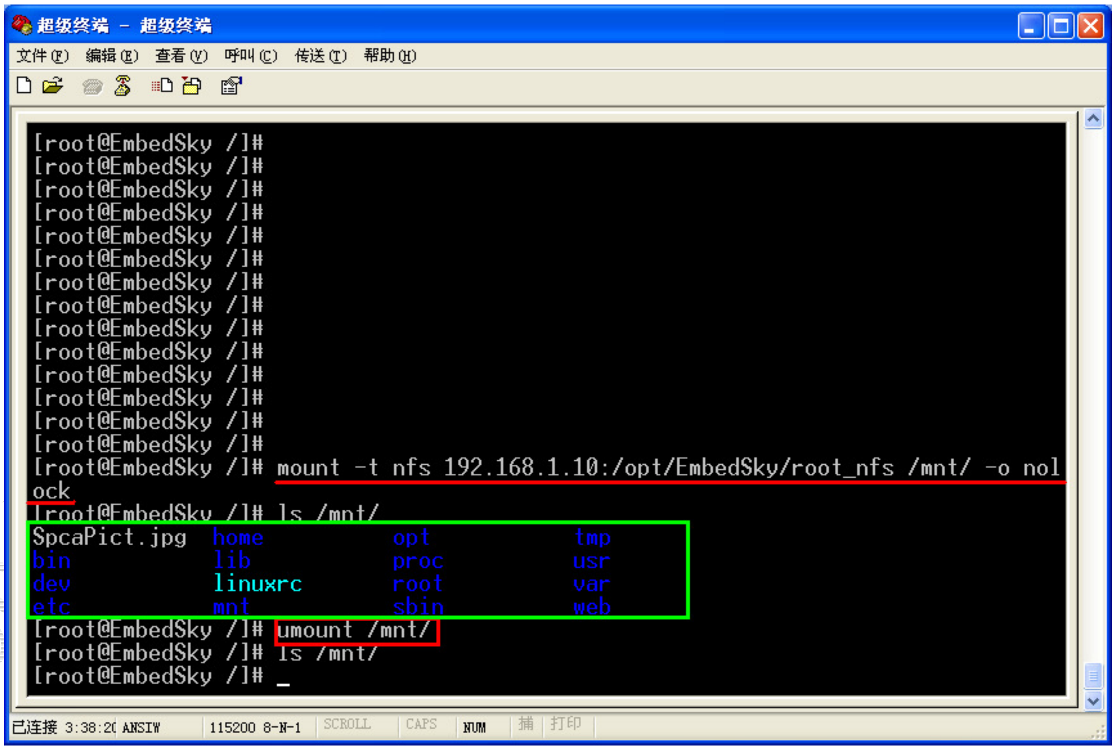


3. 6. 6 Experiment of mounting NFS

Build NFS server before the start this experiment. Set NFS server IP address: 192.168.1.10, and execute the following command to mount:

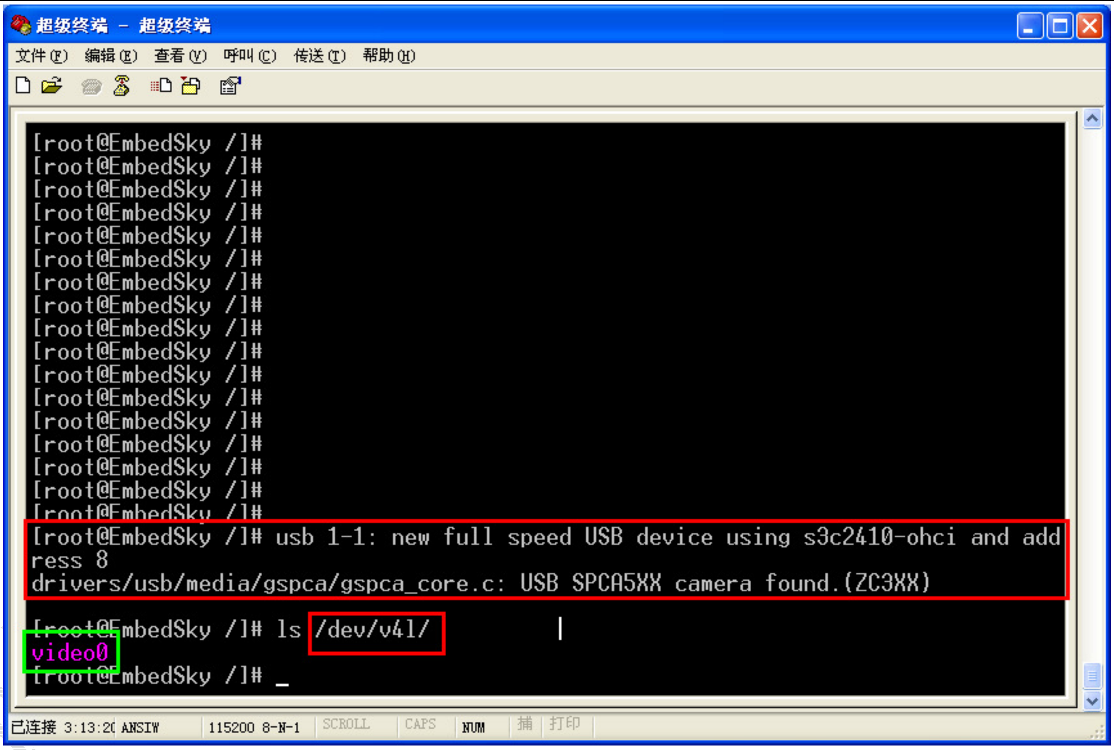
"mount -t nfs 192.168.1.10:/opt/EmbedSky/root_nfs /mnt -o nolock"

Select the NFS file under the directory "/opt/EmbedSky/root_nfs" in PC "192.168.1.10" as root file system. The following diagram appears after mount operation completes:

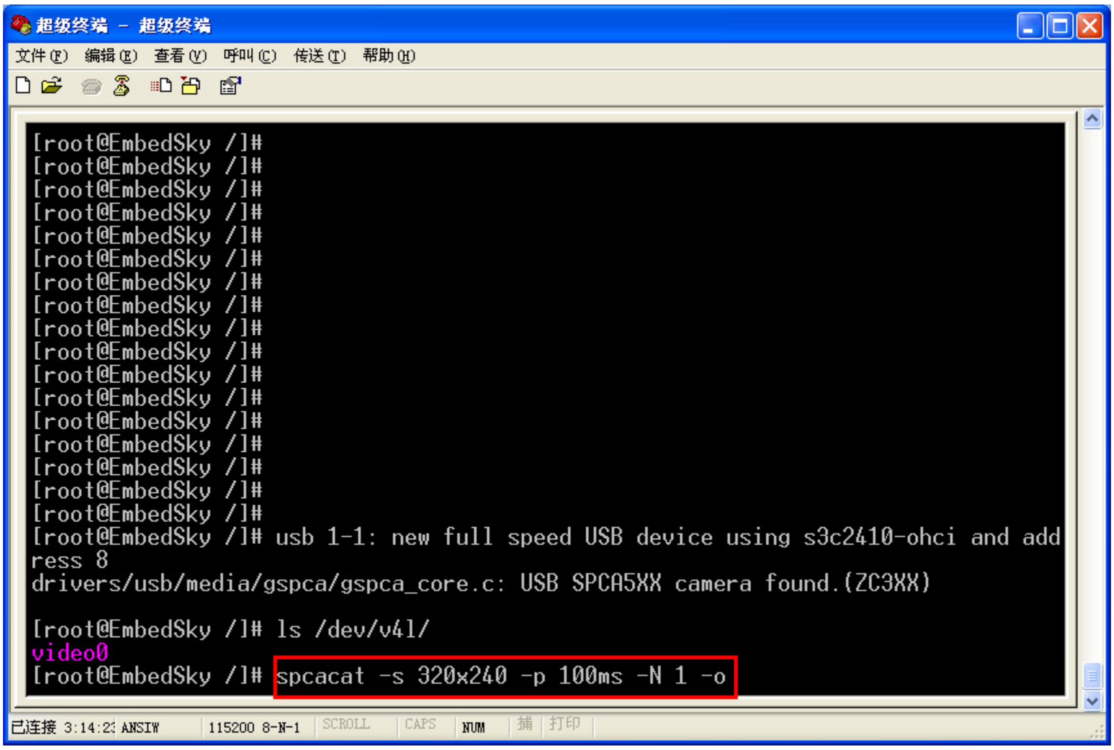


3. 6. 7 Experiment of USB camera capturing screen test

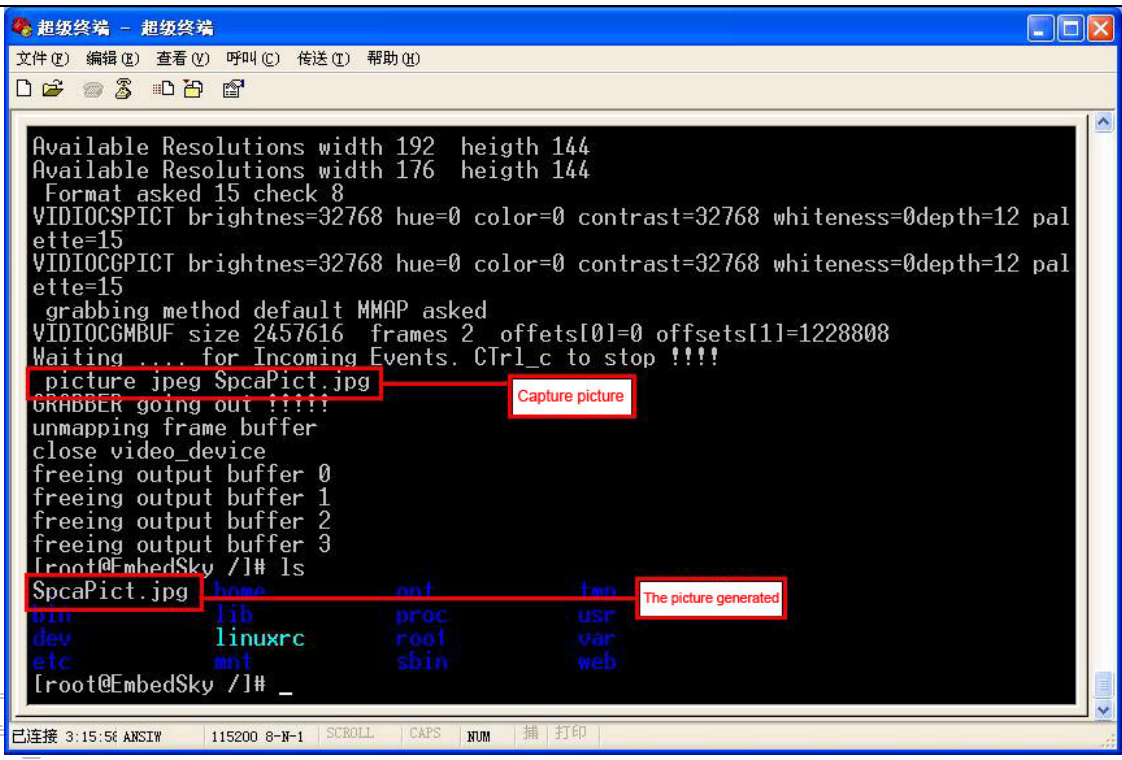
TQ2440 Development Board supports most USB cameras sold in market, for example the USB camera produced by Smics. After the camera is inserted to USB interface, the following information appears, and you can find the device name under the directory "/dev/v4l/":



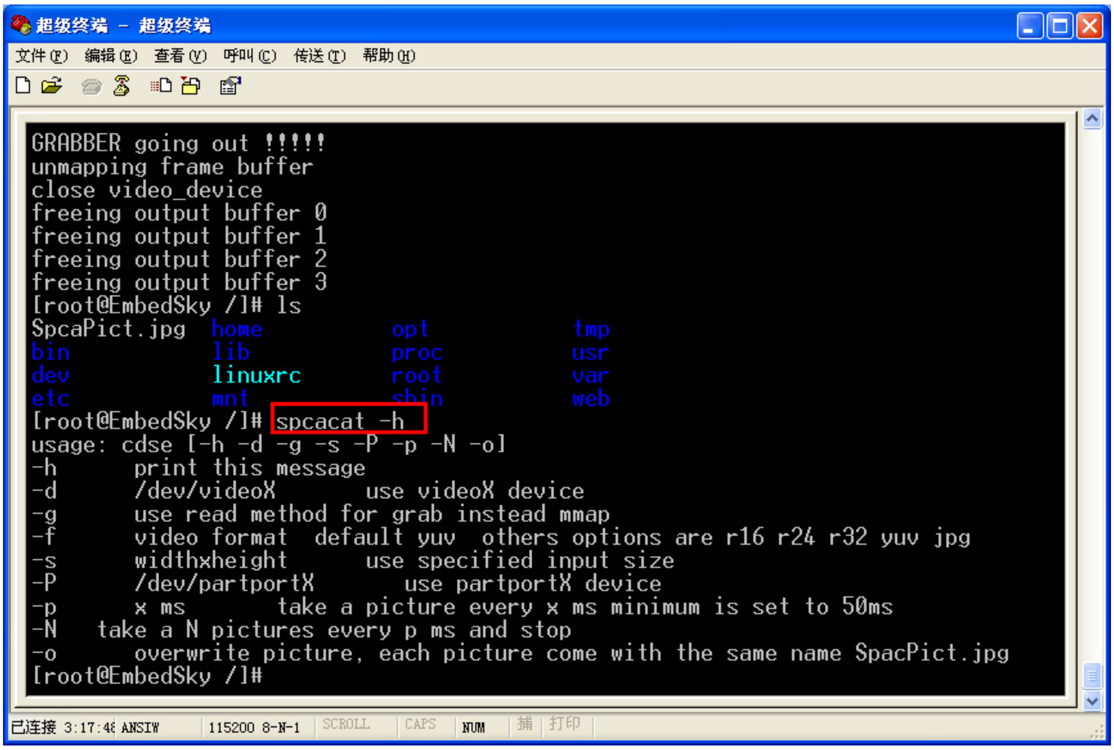
Use the software sapcacat you can get images captured by camera. Execute the command: spcacat -p 100ms -N 3



Command illustration: “-s” represents solution; “-p” represents the time interval between two captures; “-N” represents how many images to capture; “-o” represents over writing the former images named “SpcaPict.jpg” and saving the new one.

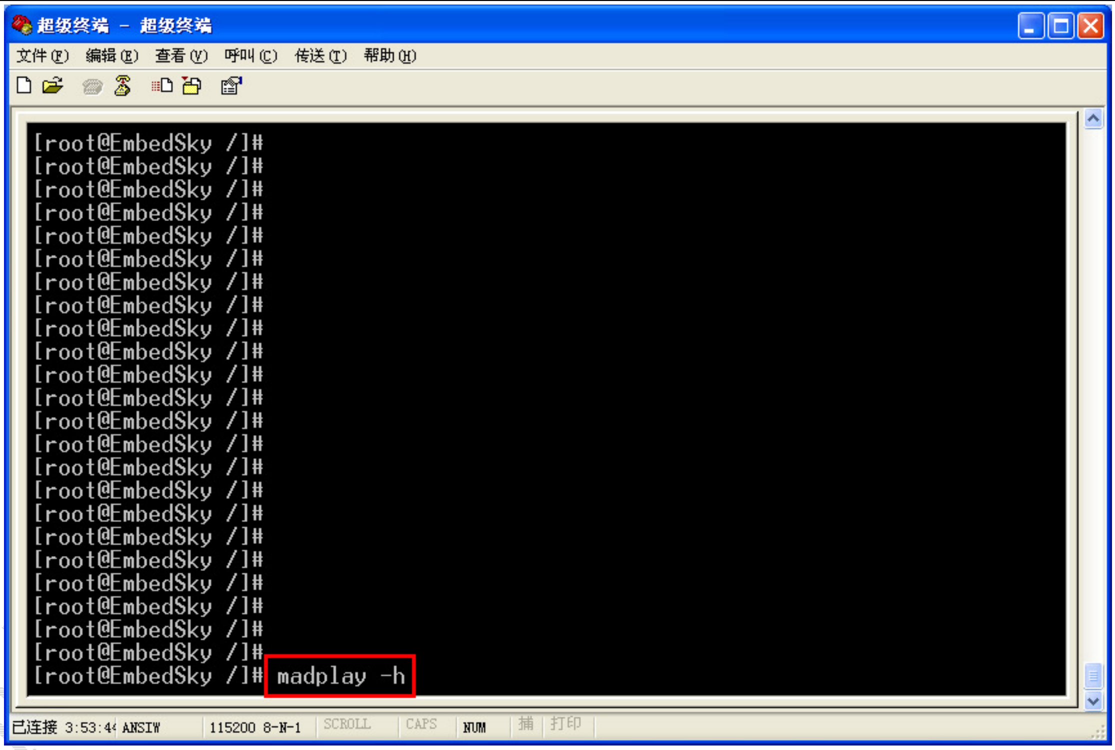


Execute the command “spcacat –h” to get more information of the parameters:

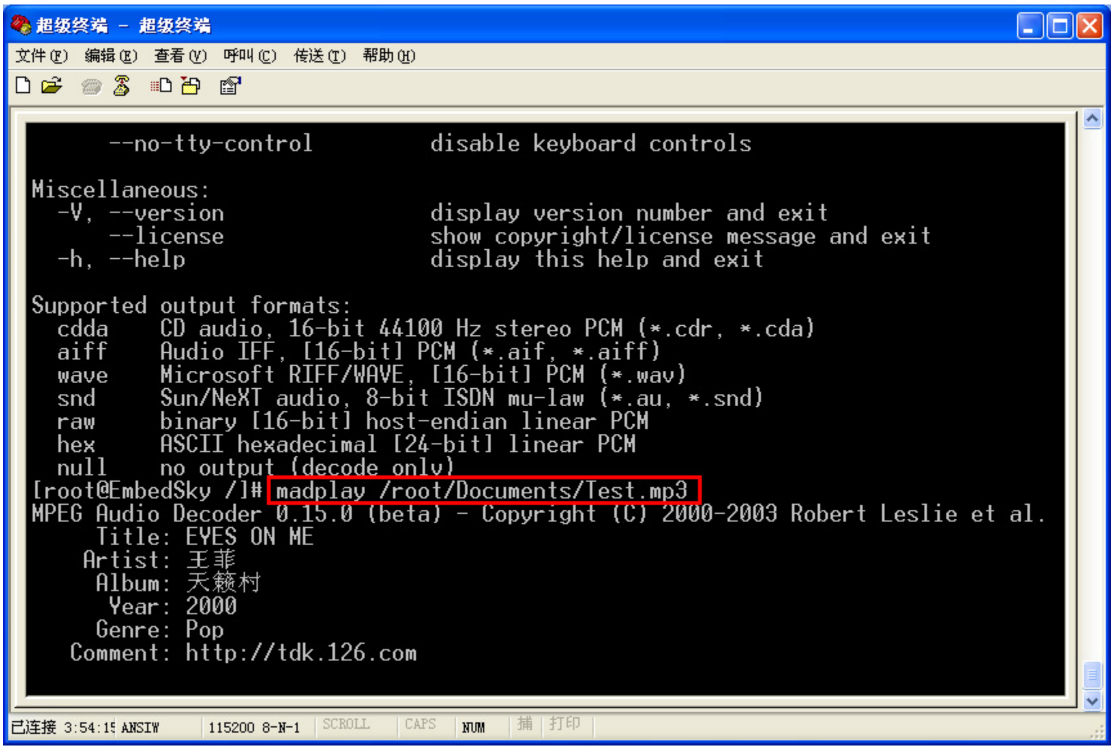


3. 6. 8 Experiment of sound card test

madplay is a MP3 player running on console, with various control modes. Execute the command “madplay -h” to get more help information:



You could execute the command "madplay xxxx.mp3" to play music in default mode. We provide a test music "madplay /root/Documents/Test.mp3" under the directory "/root/Documents/"



3. 6. 9 Experiment of files transmission with PC via serial port

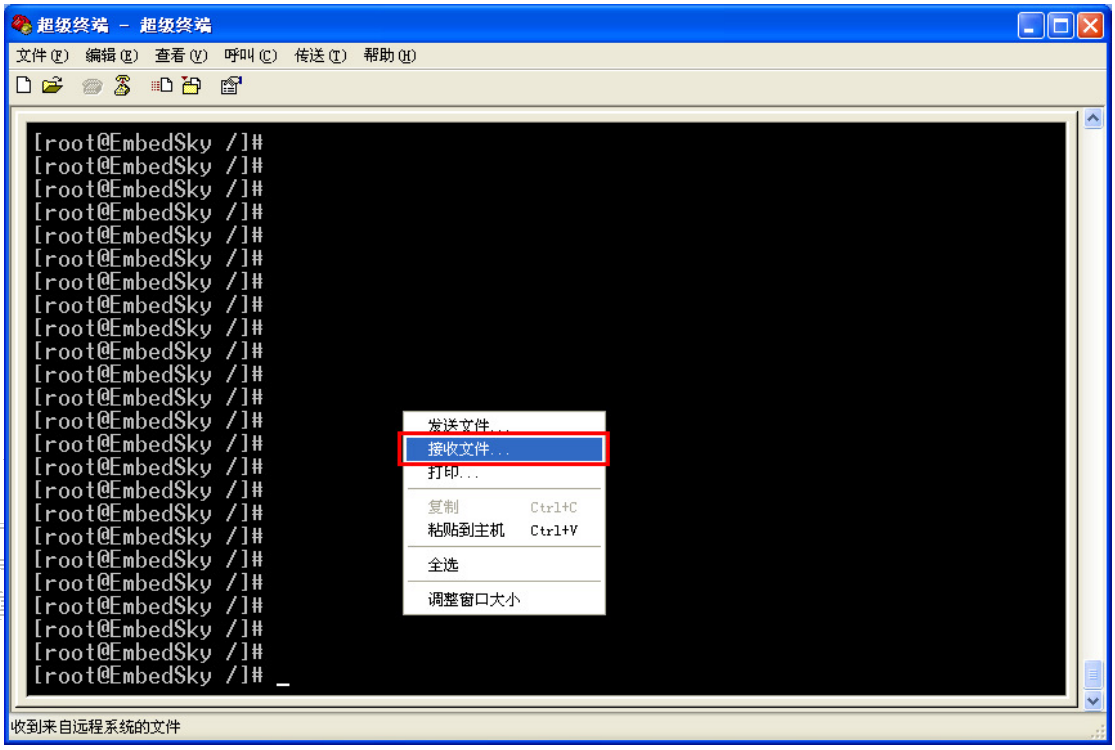
After log on OS via serial port interface, you could execute the command "rz" and "sz" to transmit files with

PC via serial port.当

The operation is introduced as follows 操作如下：

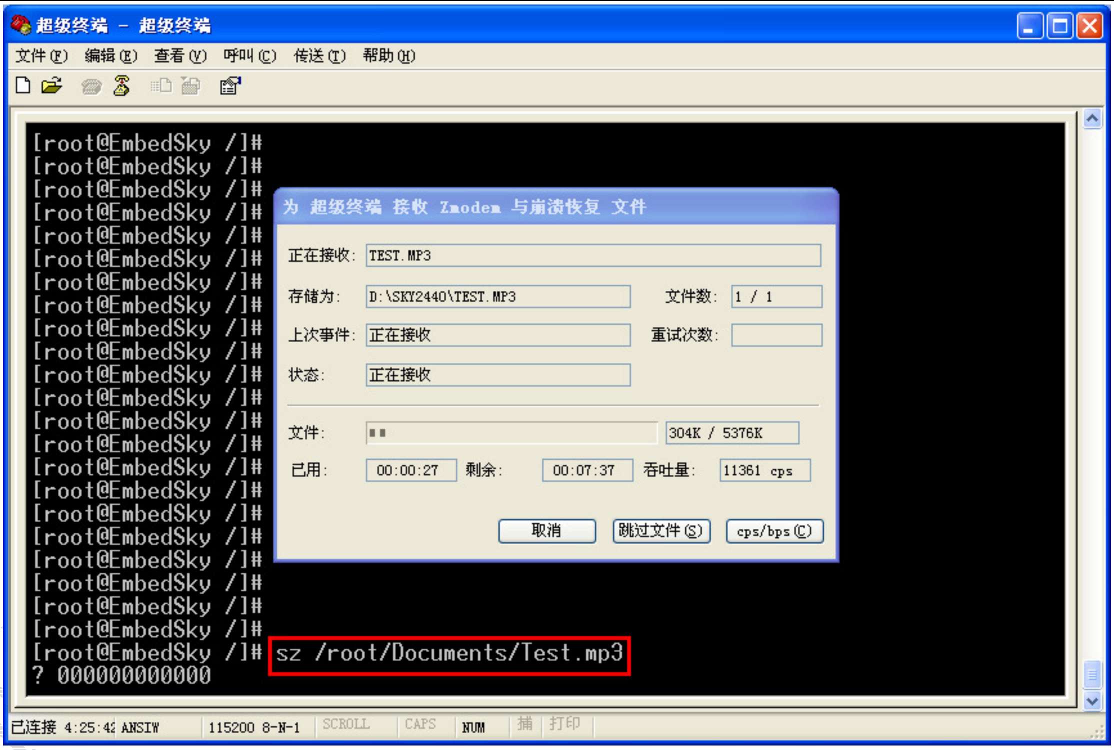
1), send files to PC:

Step1, click mouse right button in hyper-terminal interface, and select “接收文件”:

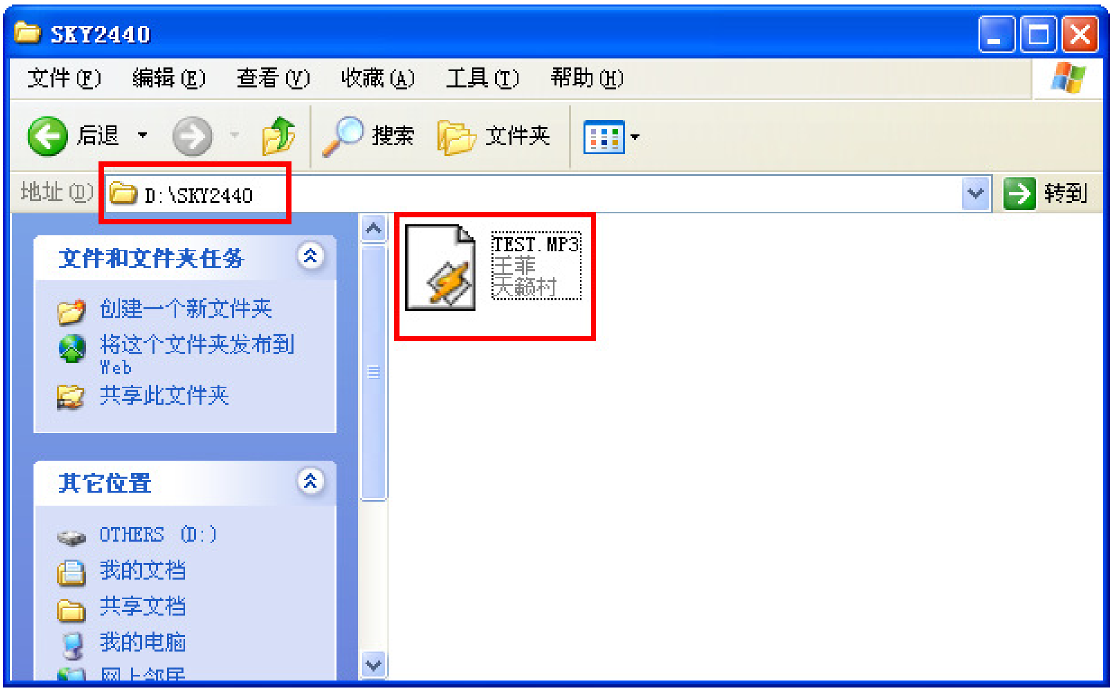


Step2, the interface “接收文件” pops up. Configure the interface as the following diagram and click “接收” to continue:

Stpe3, enter the command “sz /root/Documents/Test.mp3” in hyper-terminal to start transmitting “Test.mp3” under the directory “/root/Documents/” to PC:

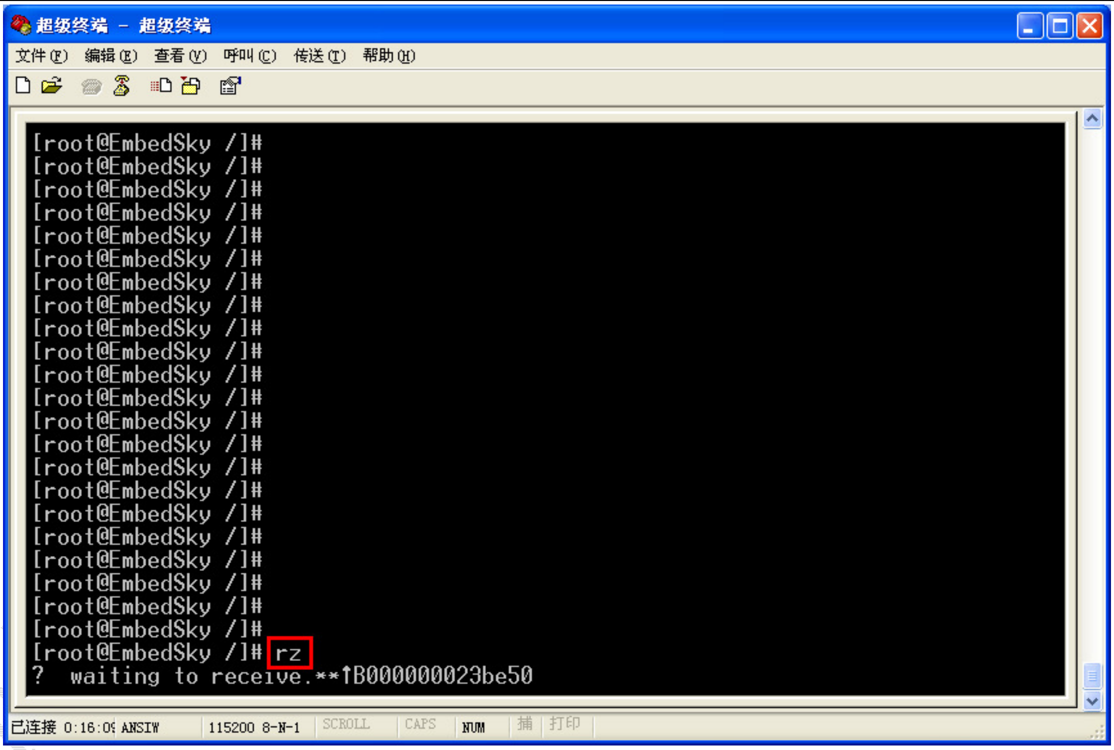


Step4, the file is automatically saved to the directory that you have just set after the transmission is over:

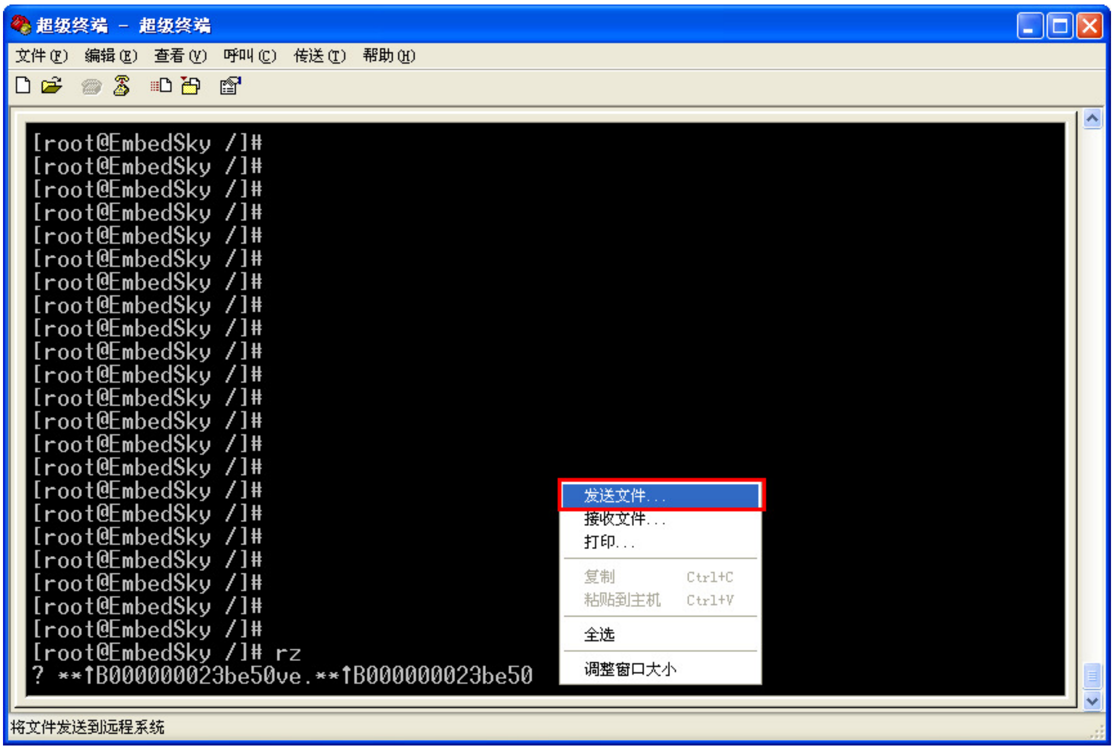


2), transmit files to platform:

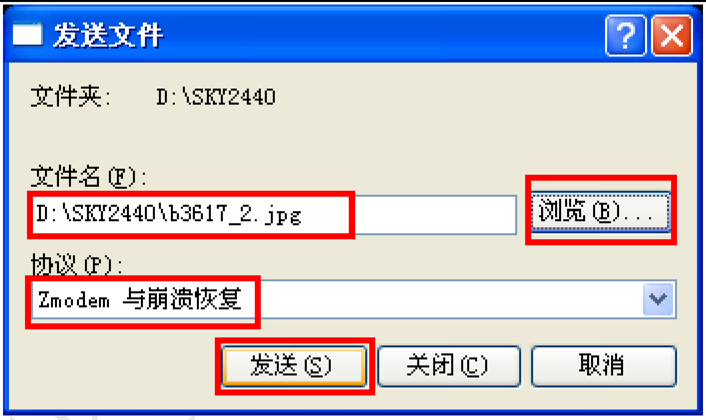
Step1, enter the command “rz” in hyper-terminal to start receiving files from PC:



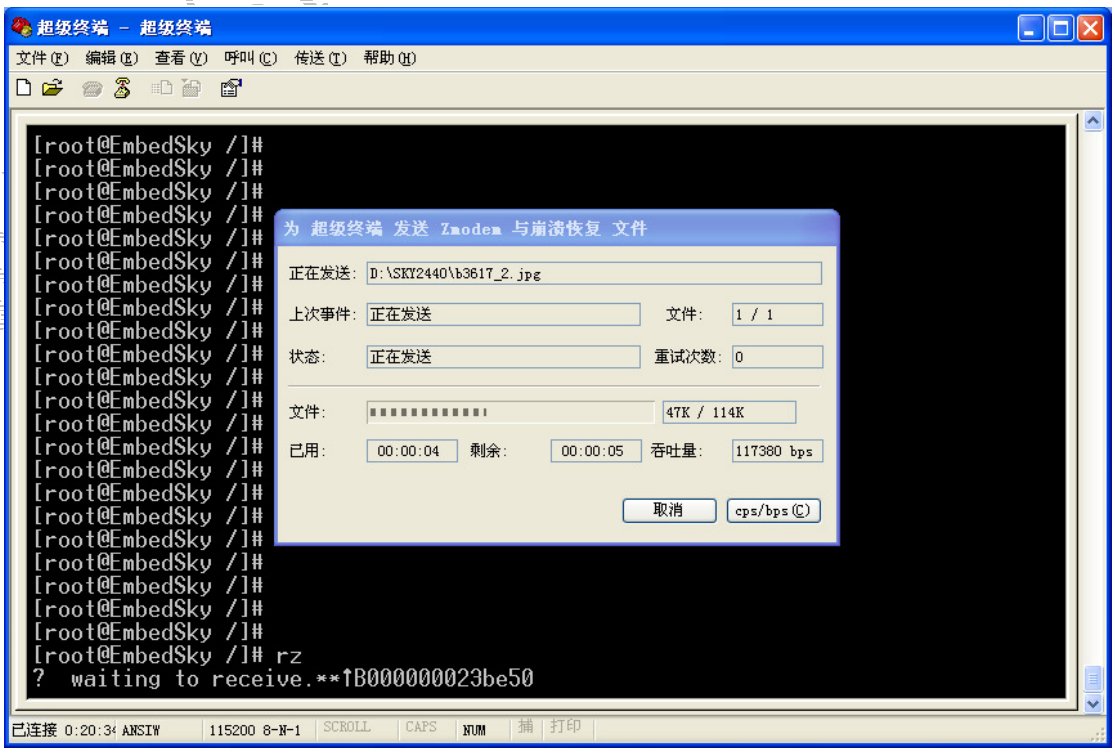
Step2, click mouse right button in hyper terminal and select “发送文件”:



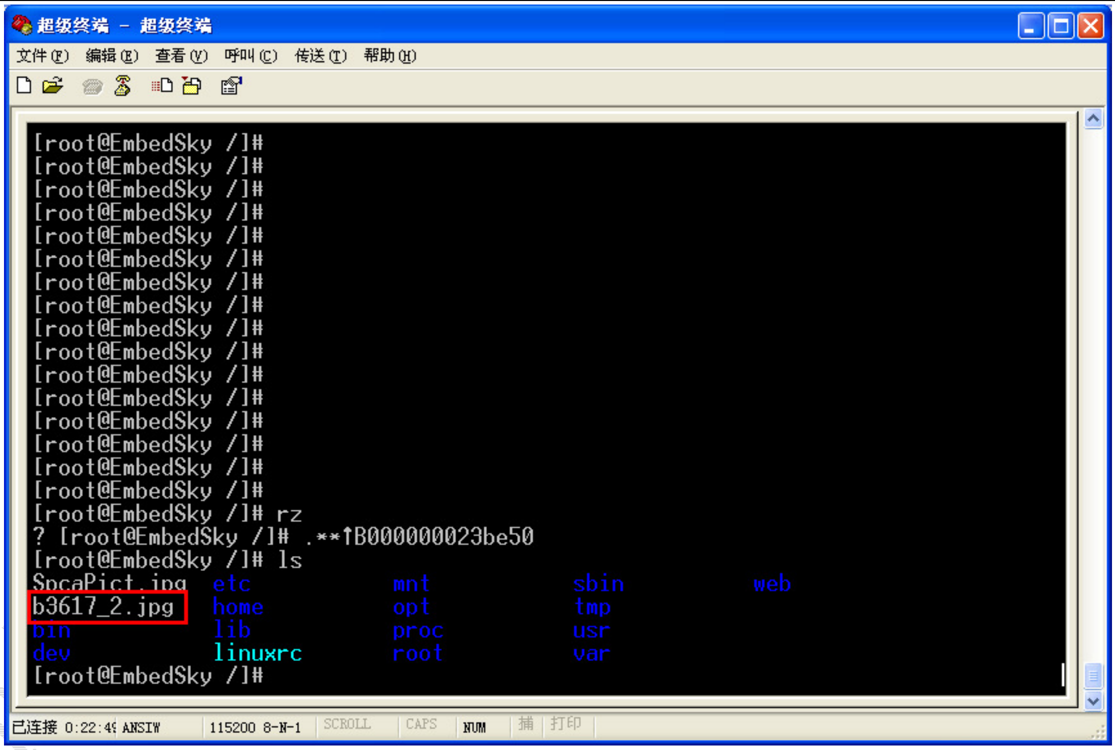
Step3, click the button “浏览” in the pop-up interface “发送文件” and locate the file for transmission, as shown in the following diagram:



Step4, click the button "发送" in the upper diagram to start transmission:

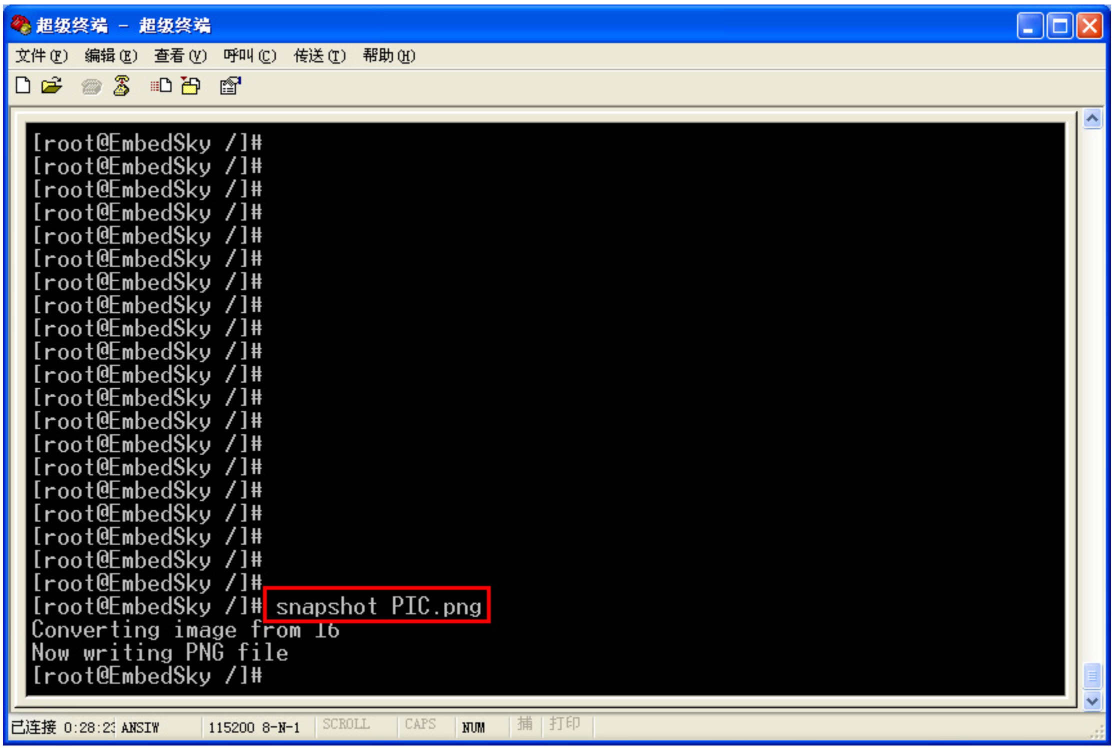


Step5, the name of the file transmitted is in the following red frame:

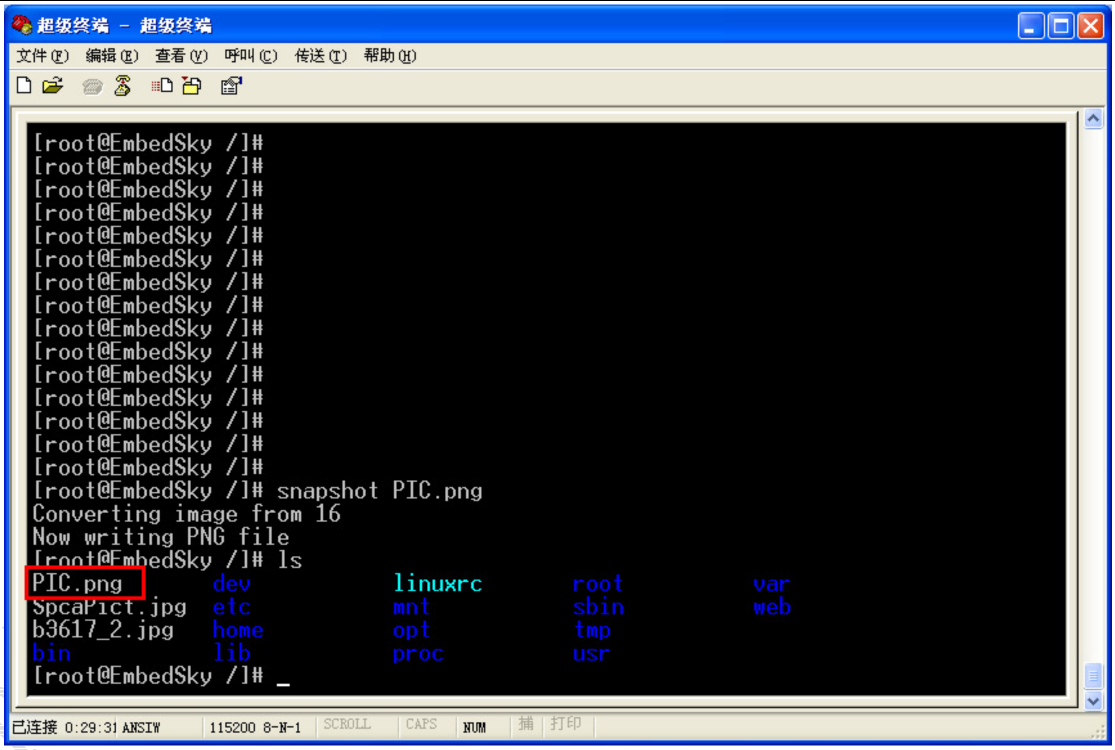


3. 6. 10 Experiment of screen capture

Execute the command “snapshot” to capture screen and save the image into the png format.
The command: snapshot PIC.png



The captured image is under the root directory:



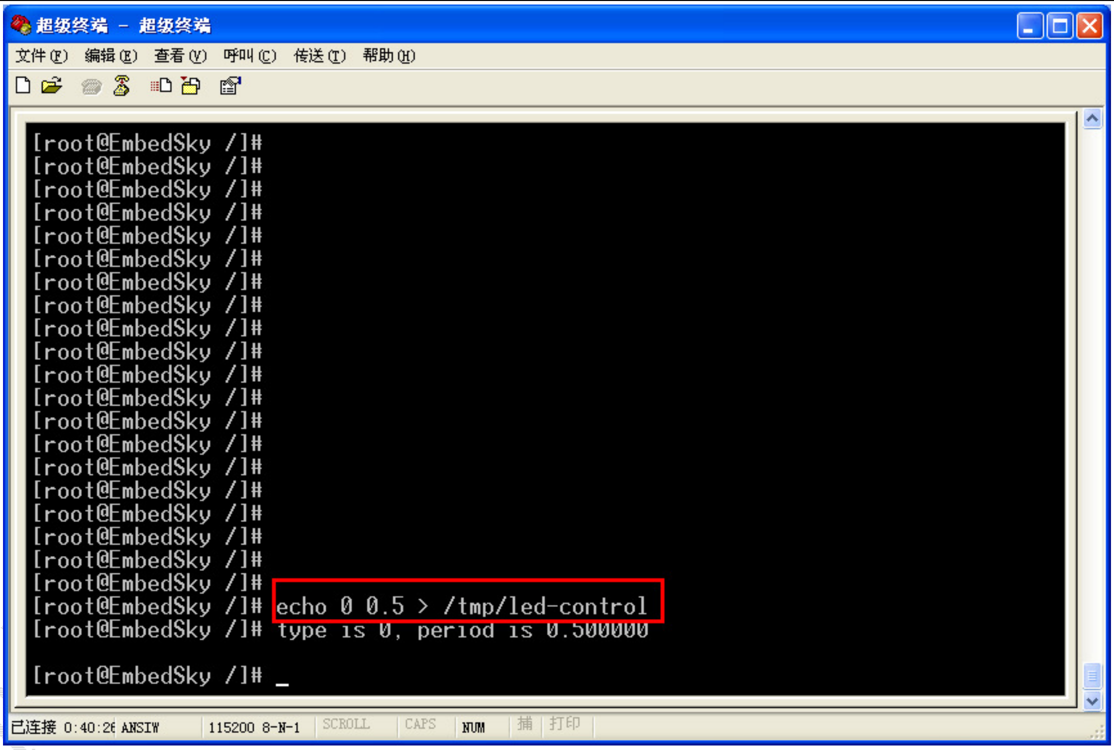
3. 6. 11 Experiment of user LED test

1), LED server

After system start-up, the user LED service application "/etc/rc.d/init.d/leds" executes automatically. The application calls a script of led-player to create a pipe file under the directory "/tmp". The user LED flash mode varies according to the parameters send to this pipe:

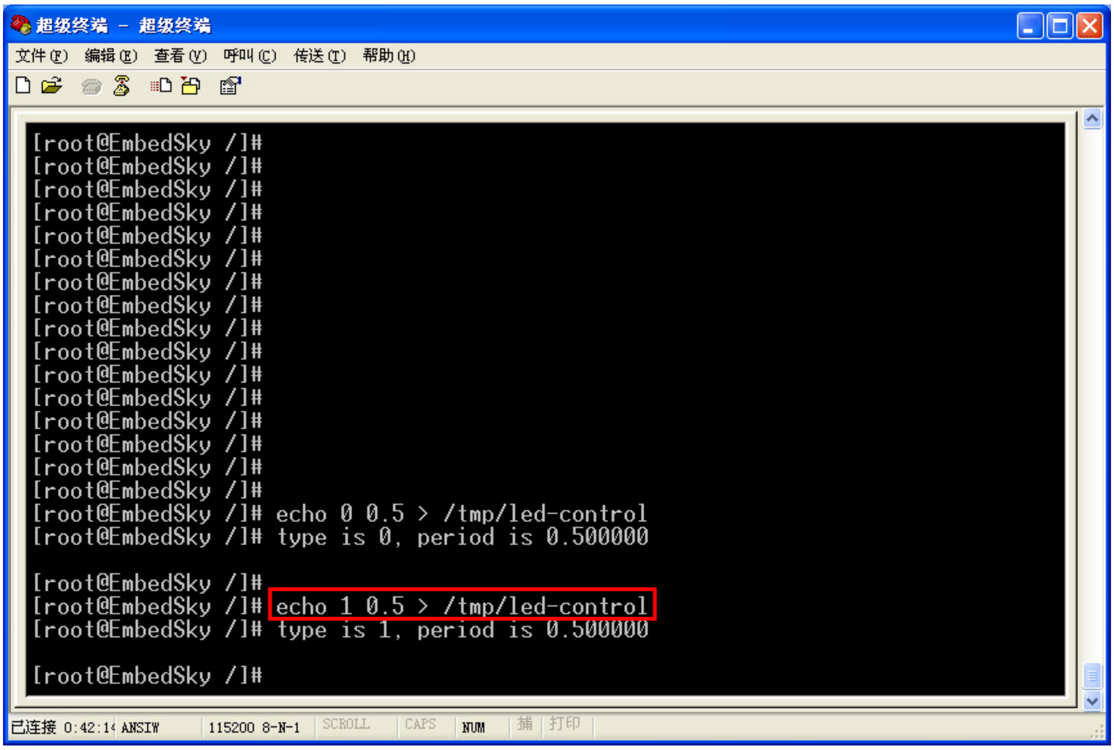
➢ #echo 0 0.5 > /tmp/led-control

The 4 user LEDs run marquee at 0.5-second interval after the command running:



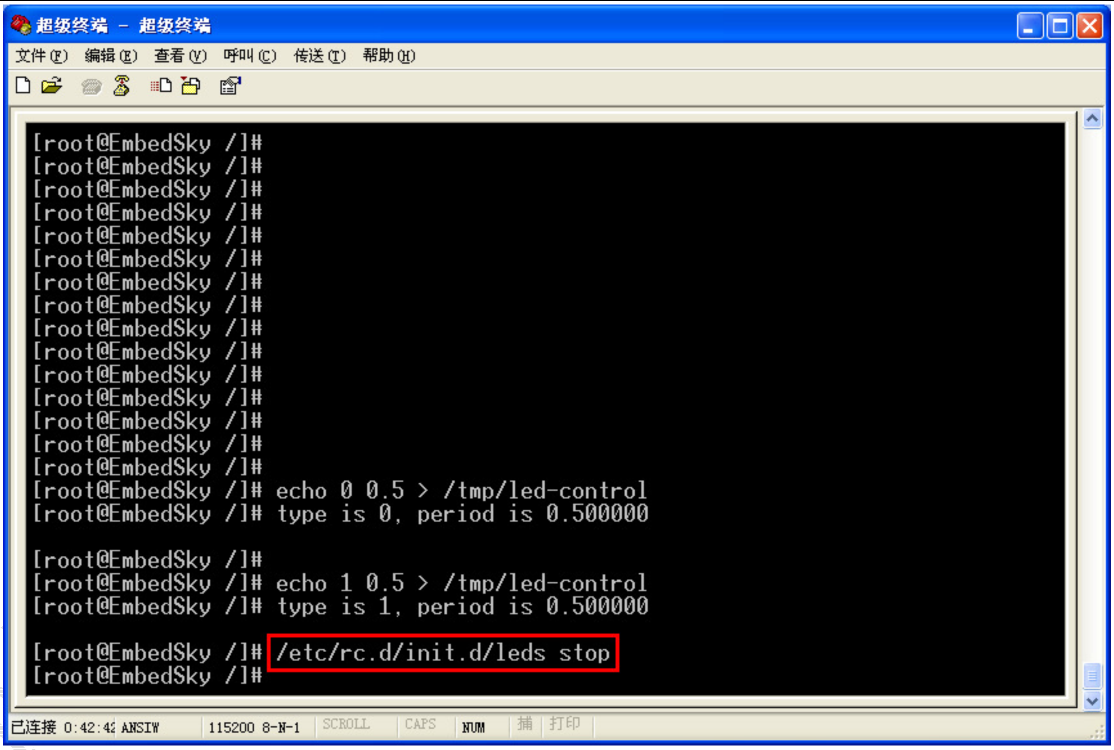
➢ #echo 1 0.5 > /tmp/led-control

The 4 user LEDs run accumulator at 0.5-second interval after the command running:



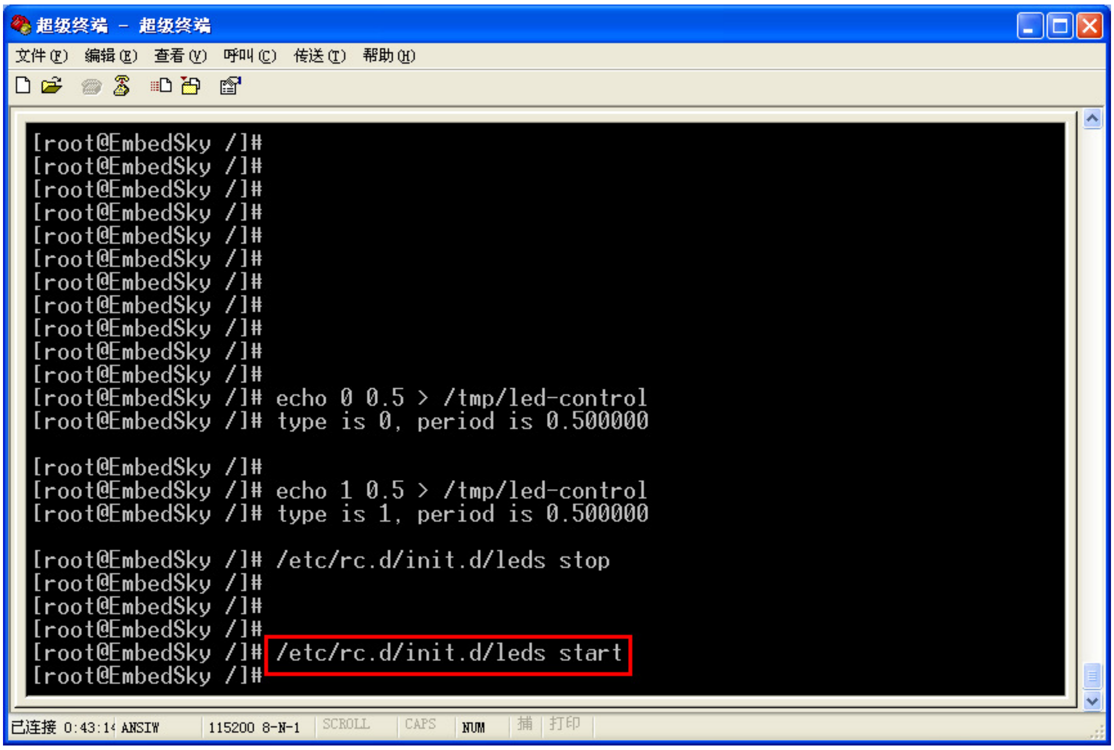
➢ #/etc/rc.d/init.d/leds stop

The 4 user LEDs stop flashing after the command running:



➢ #/etc/rc.d/init.d/leds start

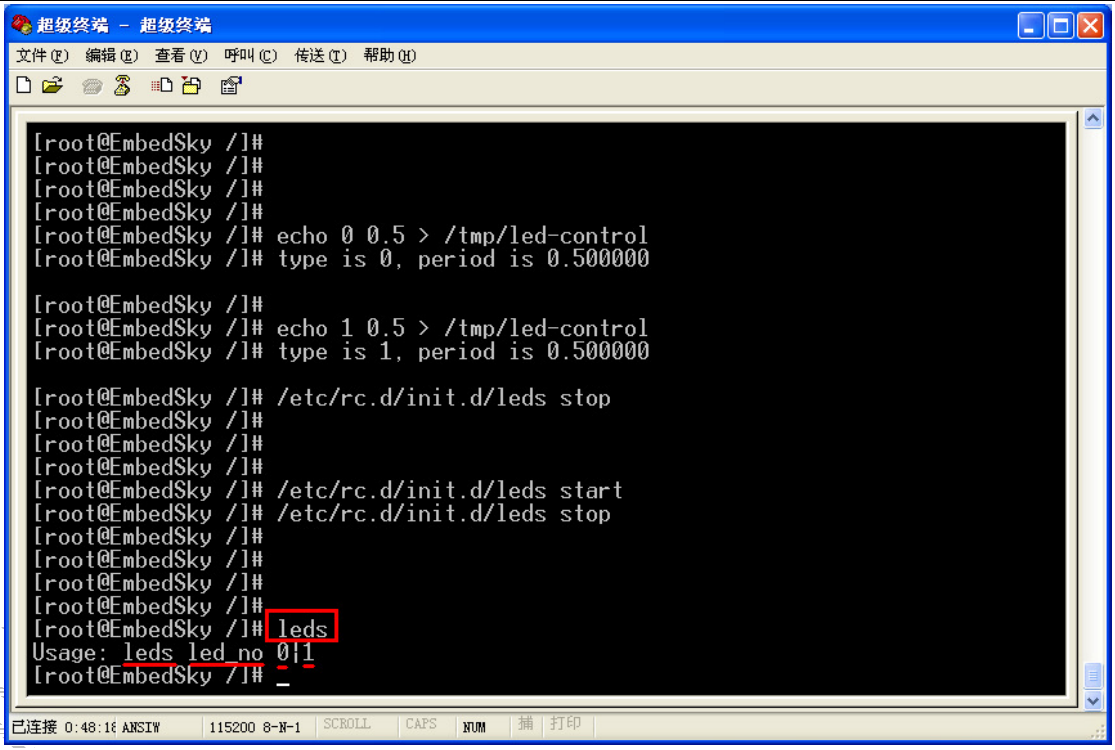
The 4 user LEDs re-flash after the command running:



2), control LED separately

The application “/sbin/leds” can control LED separately. You must stop led-player before run this application. Execute the command “/etc/rc.d/init.d/leds stop” to stop led-player.

“leds” usage method:



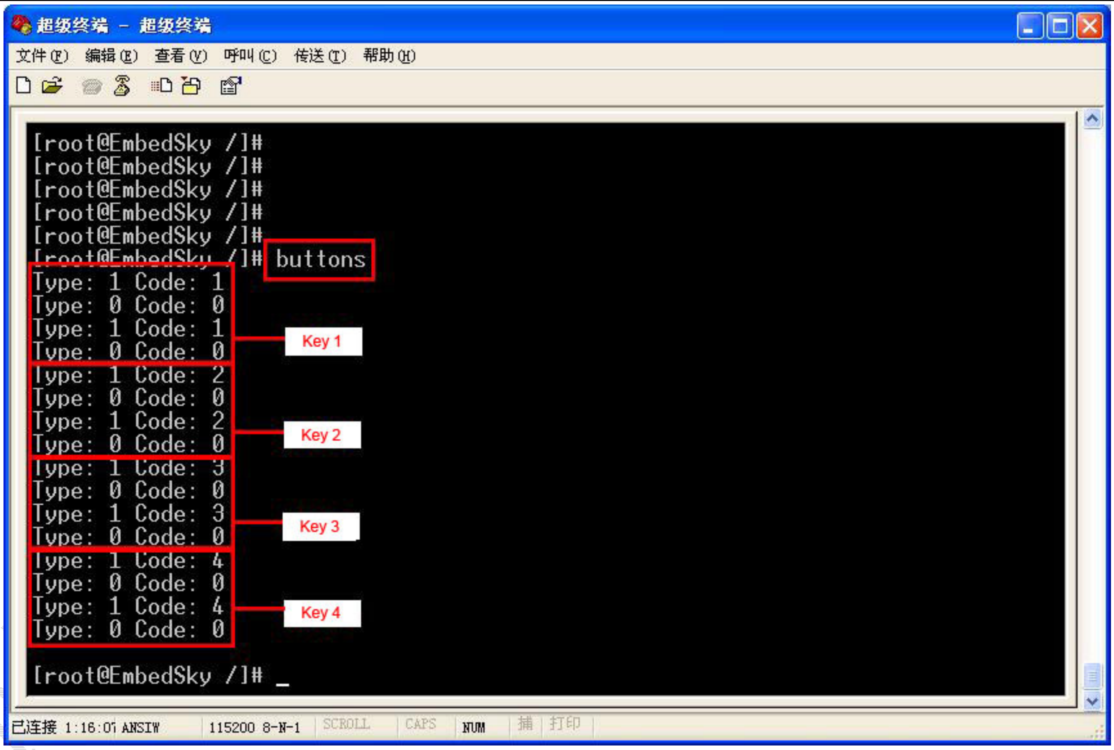
"led_no" is the LED sequence number (0, 1, 2, 3); The value "0" and "1" represent operation switching off and switching on.

For example "leds 3 1" means switch on LED3.

3. 6. 12 Experiment of user keyboard test

Mount user keyboard driver first:

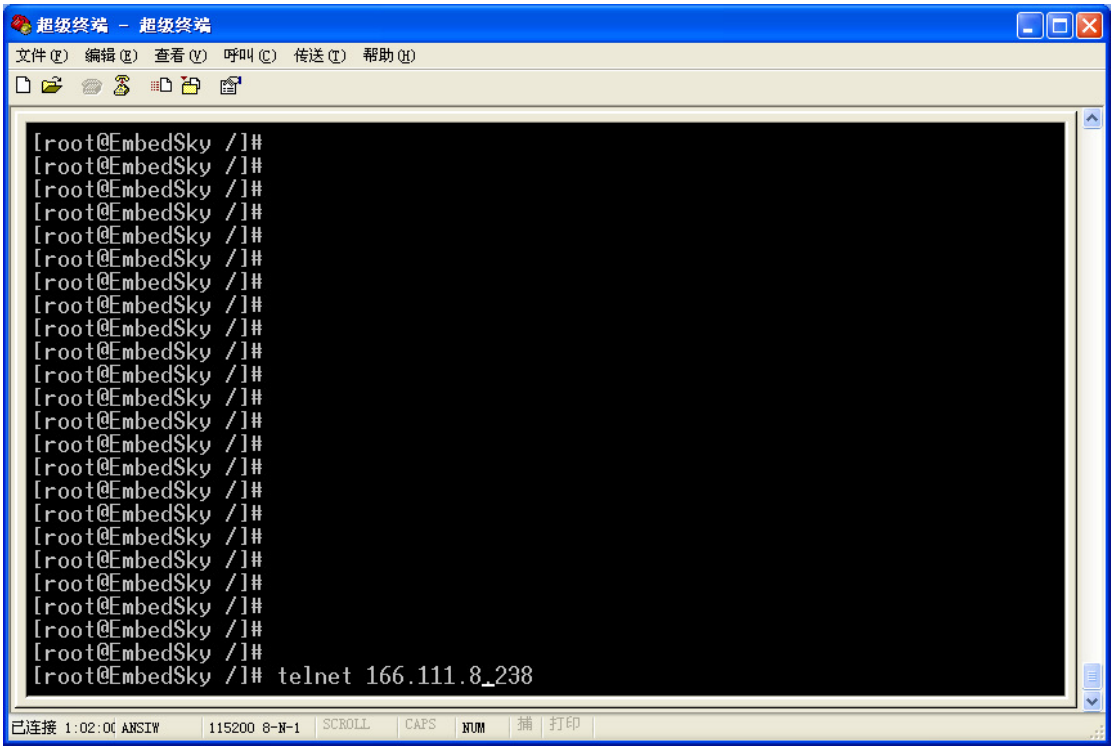
Enter "buttons" to start keyboard test. The following diagram displays the responses when pressing the 4 buttons separately:



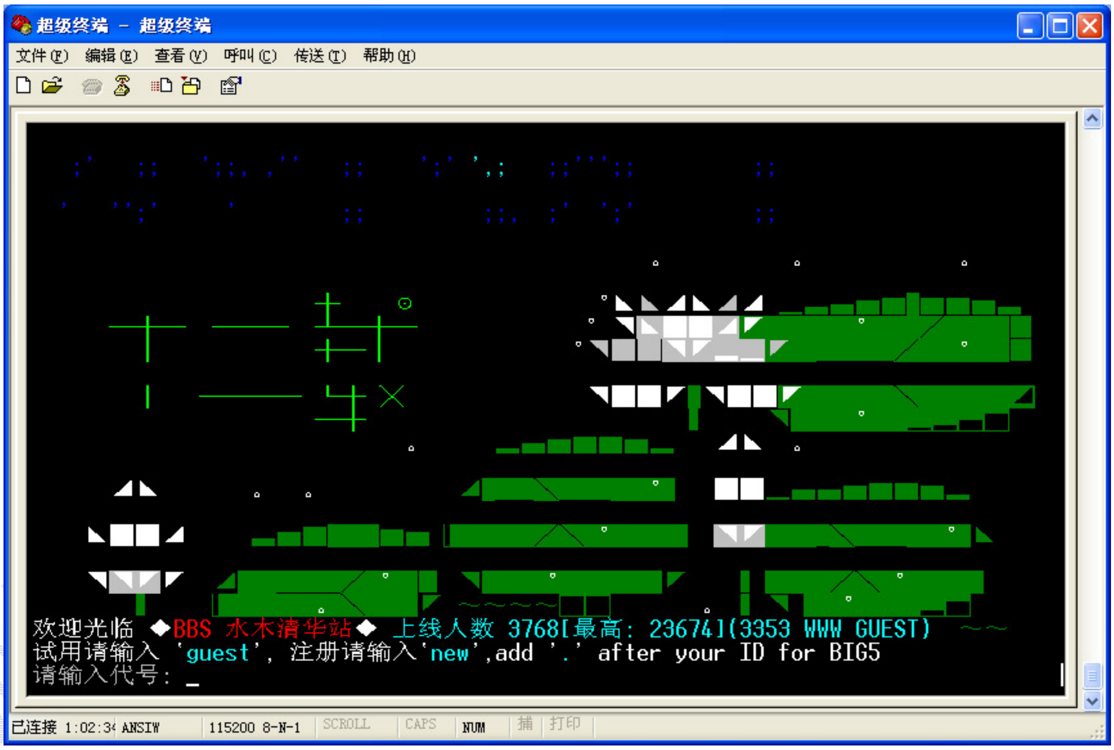
3. 6. 13 Experiment of log in BBS in telnet

telnet is a kind of commonly used remote login tool. We can use telnet to log in other telnet servers from the platform. If the platform is connected to internet, you could use telnet command to log in external BBS.

Use telnet to log in BBS “水木清华”:

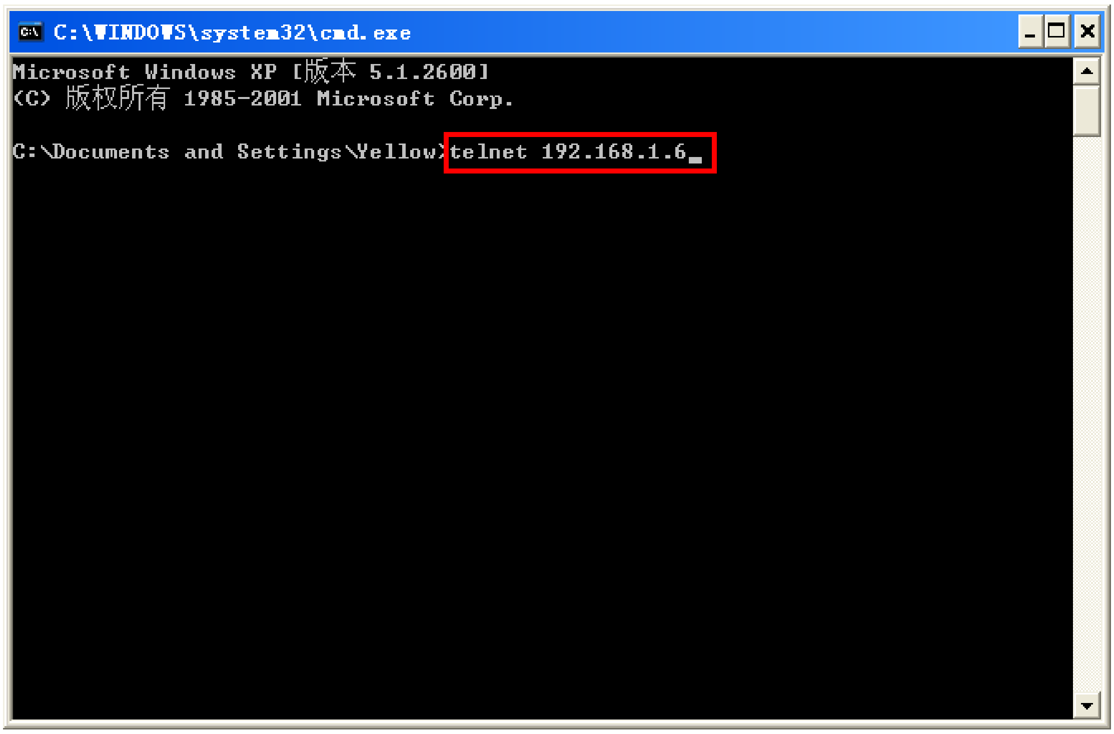


Log in successfully:

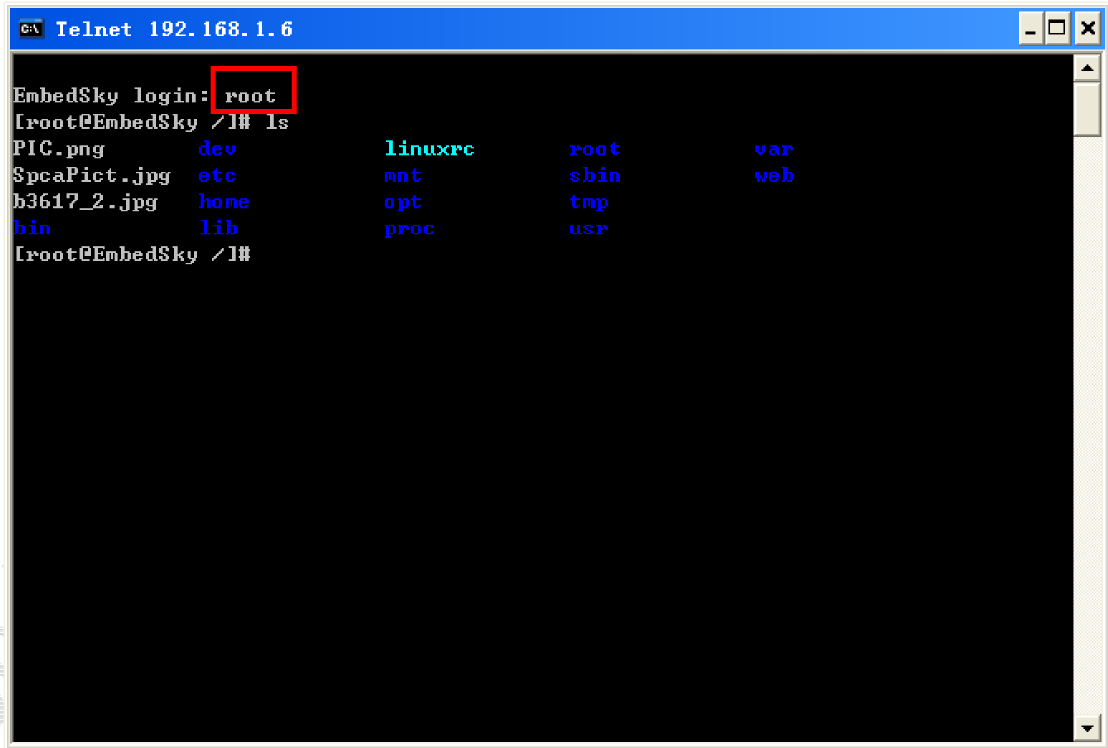


3. 6. 14 Experiment of remote login platform by telnet

The user can directly log in platform after system starts up. Enter the command “telnet 192.168.1.6” in command window in Windows OS, and press return-key to continue:



Enter “root” in the following log-in interface to enter into the system:



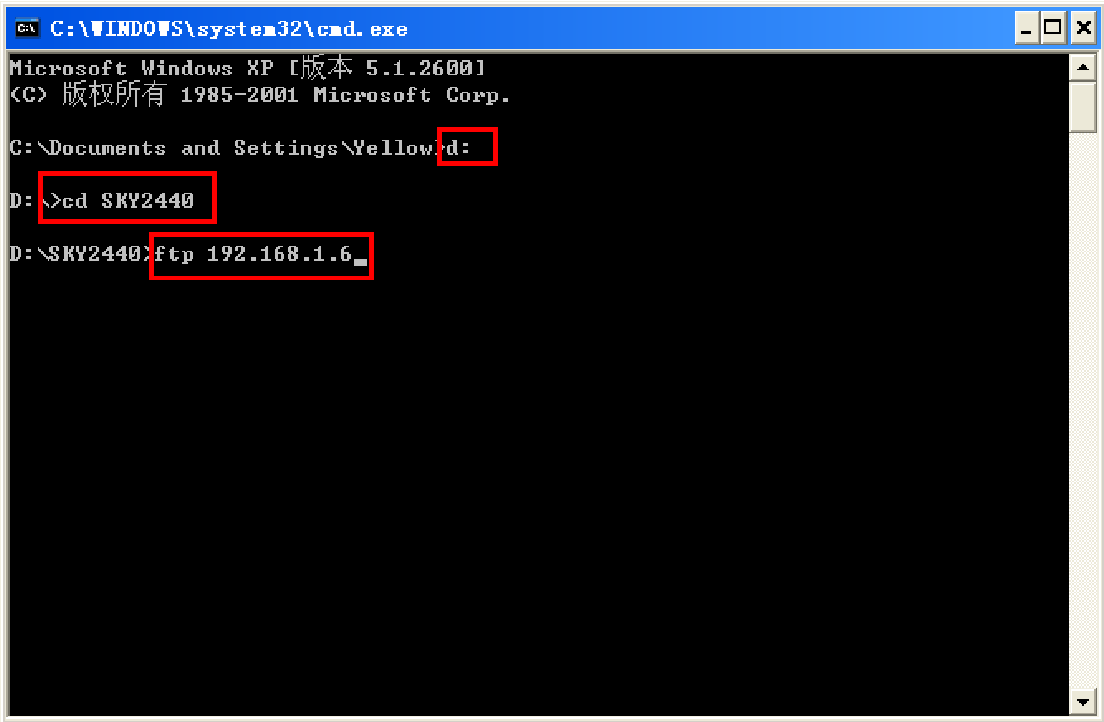
3. 6. 15 Experiment of FTP remote file transmission

We can use the ftp application contained in Linux or Windows to log in remote host and transmit file, if the remote host support ftp service and the authority is available. TQ2440 provides ftp application and ftp service. Here we make a test to log in platform from PC command window and send files to the platform.

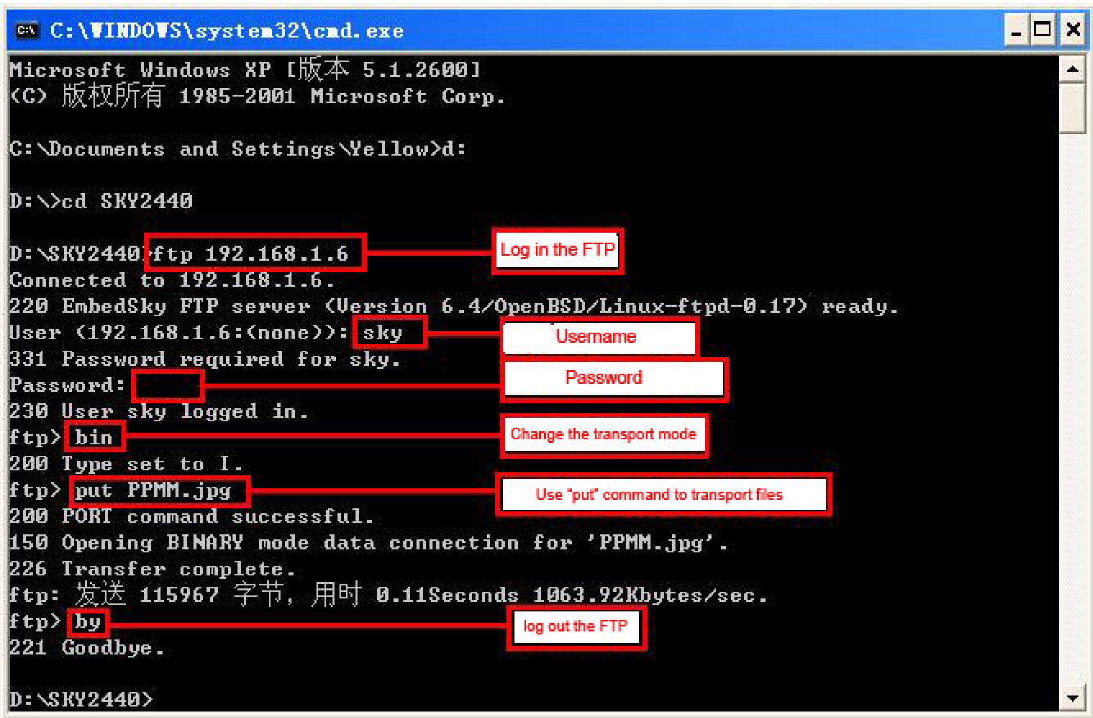
Caution: Make sure the transmission file is under the same directory with ftp and it is available. Here we use the file “PPMM.jpg” for transmission.

After the transmission is over, the file “PPMM.jpg” is added to the directory “/home/sky/”.

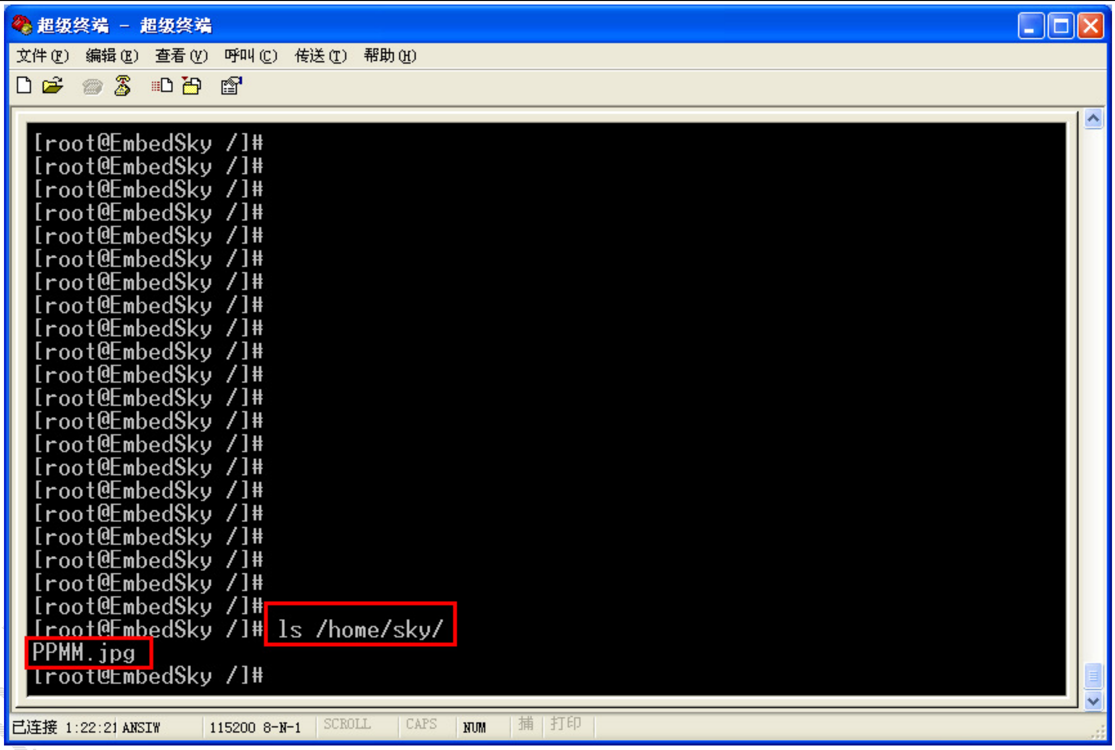
Enter “ftp 192.168.1.6” in command window and press return-key to continue:



Operating as the following diagram:



Enter the command “ls /home/sky/”, you can find the file “PPMM.jpg”



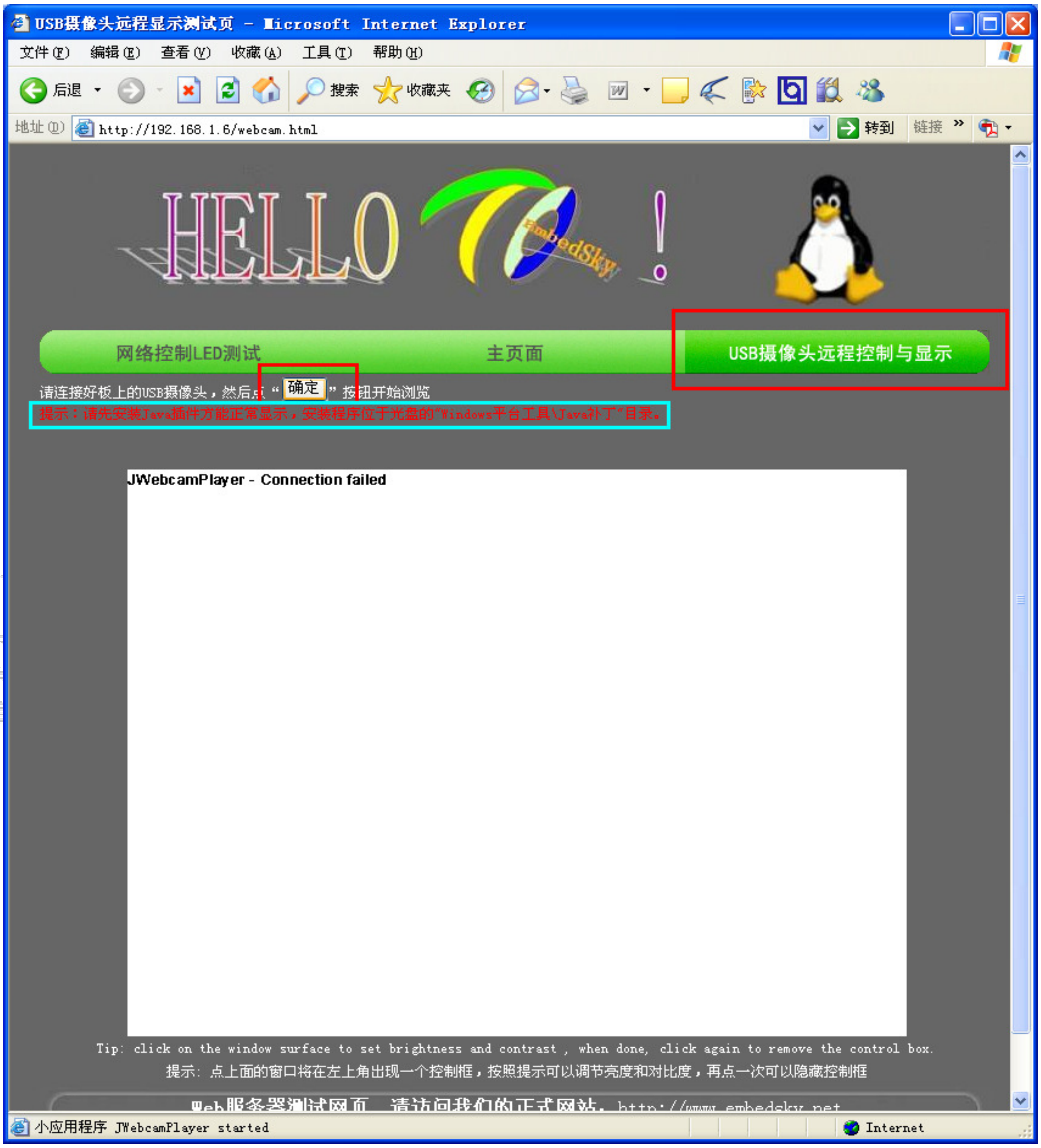
3. 6. 16 Experiment of Web server test

The Web server runs automatically after system start-up. The user can use webpage explorer to access the webpage of Web server based on platform. Enter “192.168.1.6” in address table and press return-key. Then you can access the following page:

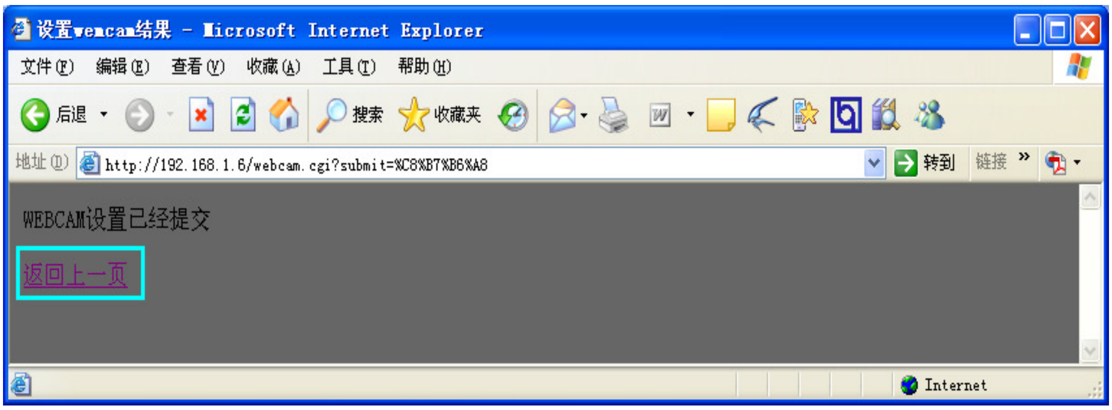
3. 6. 17 Experiment of USB camera remote control

Click the option “USB 摄像头远程控制与显示” in the upper webpage to access USB camera test page:

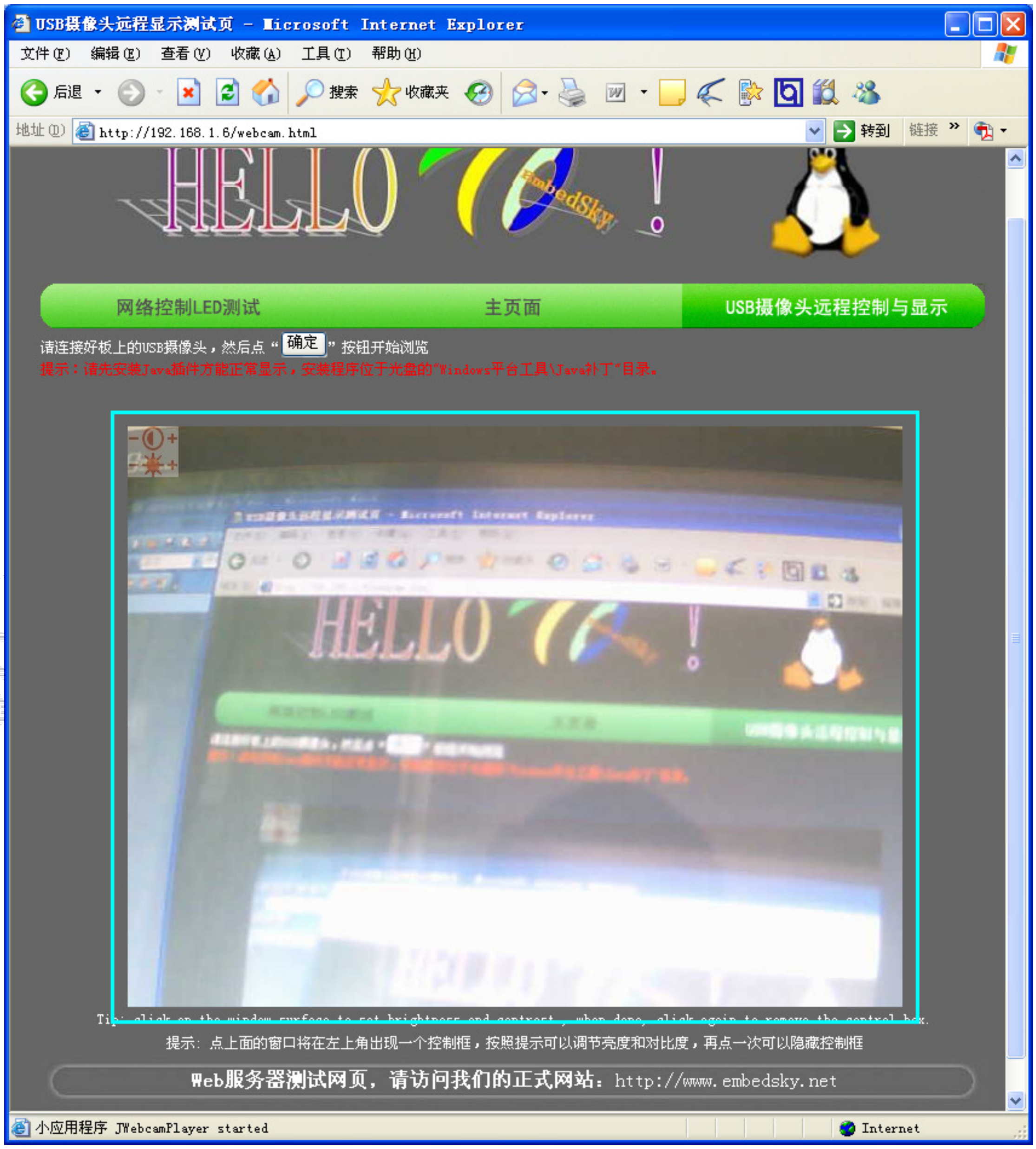
(caution: Make sure the Jave patch has been installed successfully which is under the directory “Windows 平台工具/Java 补丁”. And it is suggested to use IE explorer provided by Windows, otherwise the experiment will probably fail)



Click "确定" in the following page:

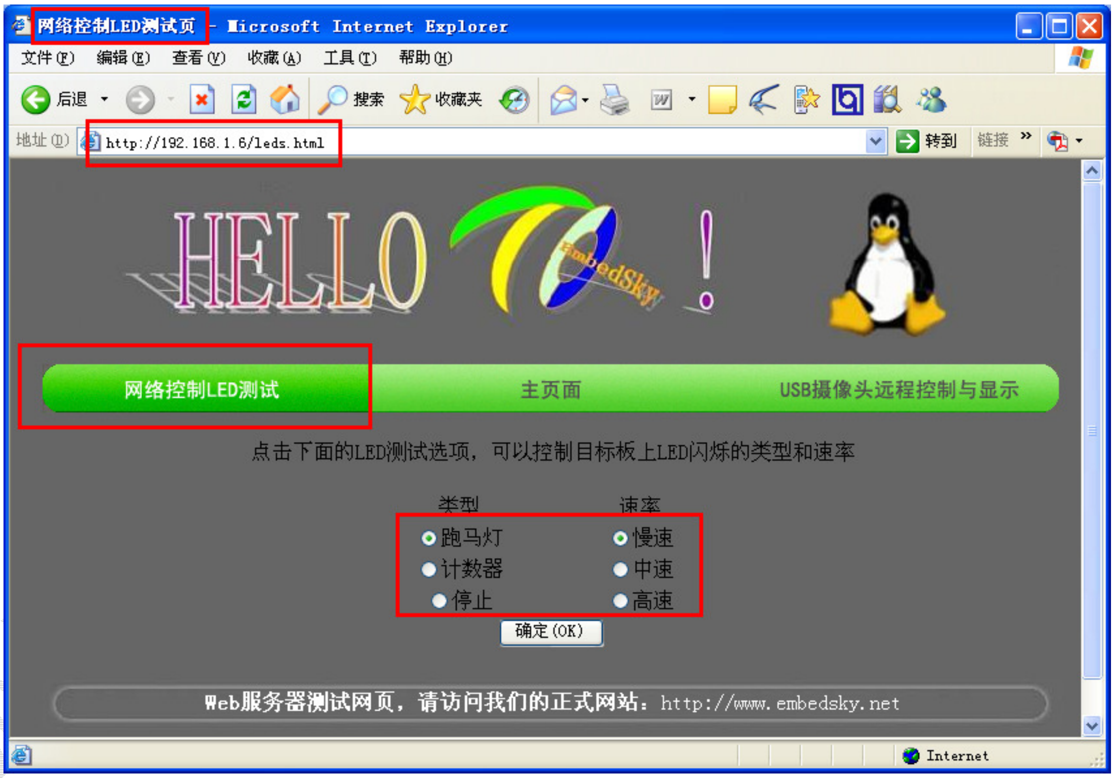


Click "返回" to go back to the former interface. The dynamic images appear in webpage now as shown in the following diagram:



3. 6. 18 Experiment of user LED remote control

Click "网络控制 LED 控制" to access LED control page:



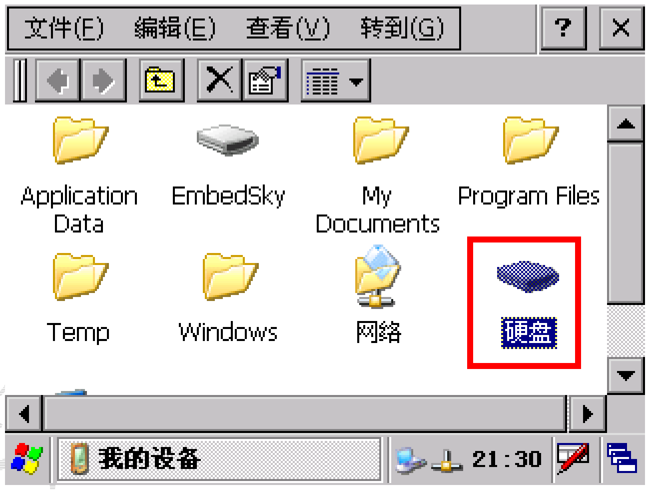
Select the options “类型” and “速率” and click “确定(OK)”. Then the user LEDs on platform start flashing.

3. 7 WinCE experiment

Caution: The IP address of 100M network card is “192.168.1.6”.

3. 7. 1 Experiment of USB mobile storage device test

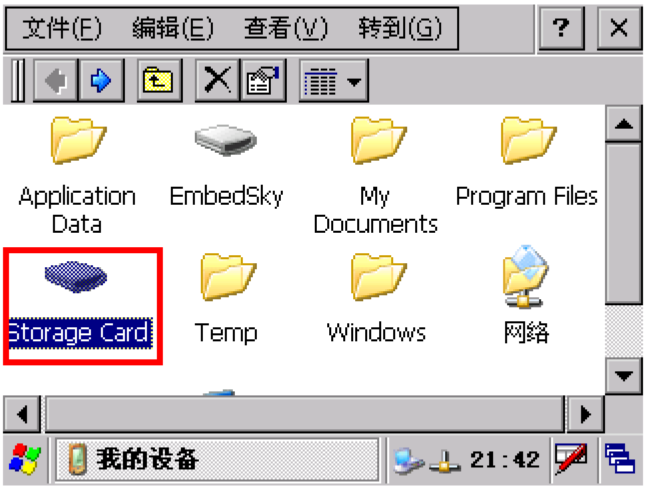
Like using USB mobile storage device in Windows XP system, insert the USB device into USB Host interface in WinCE system, and the system will load the device automatically after a few seconds. Double-click the icon “我的电脑” on desktop, and open resource manager. You can find the USB mobile storage device disk “硬盘”, as shown in the following diagram:



Double-click the icon "硬盘" to access the USB mobile storage device:

3. 7. 2 Experiment of SD card test

After SD card has been inserted into SD interface on platform, the SD card disk "Storage Card" could be found in resource manager of platform:



Double-click the icon to access the SD card:

3. 7. 3 Experiment of Flash power failure protection

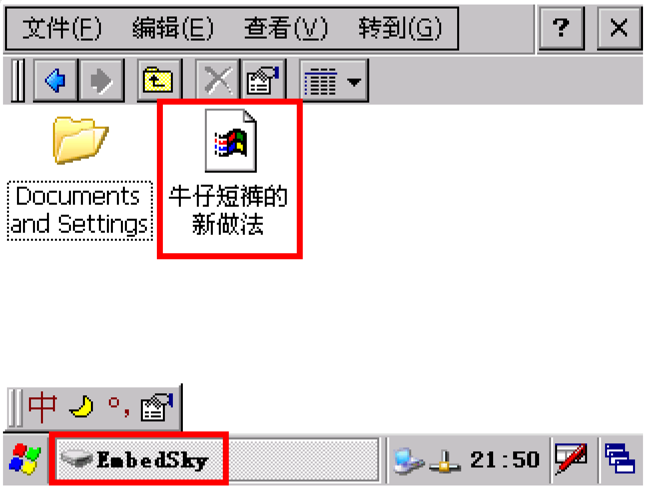
The free space in Flash could be used as disk which is supported by BSP package in WinCE. This part of Flash can be used for power failure protection, and it exists in WinCE in the form of the directory “EmbedSky”.

Test the function of “EmbedSky”:

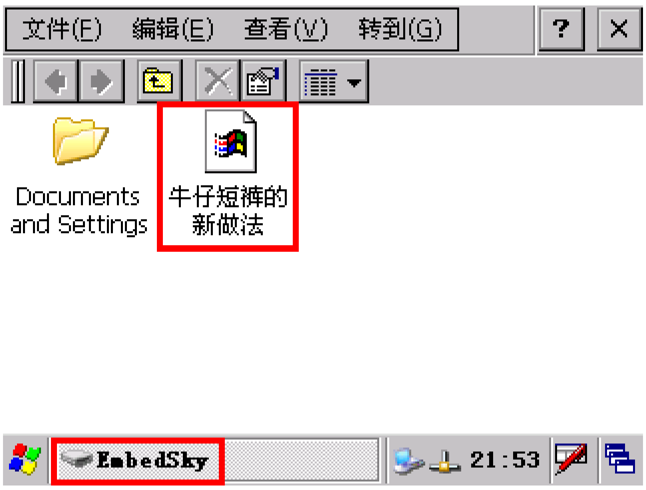
Double-click the icon “我的电脑” and find “EmbedSky”:

Double-click to access it.

Copy the file “牛仔短裤的新做法” from SD card to here:



The file still exists after system restarting:



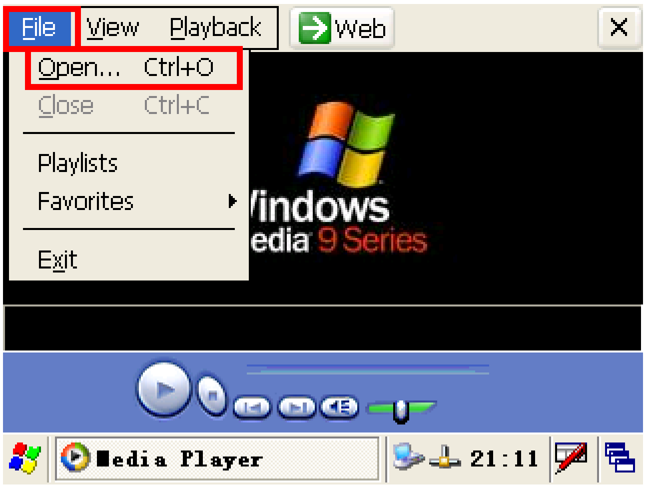
3. 7. 4 Experiment of player utilization

WinCE supports two kinds of player: The one is “Media Player”, and the other is “超级播放器”.

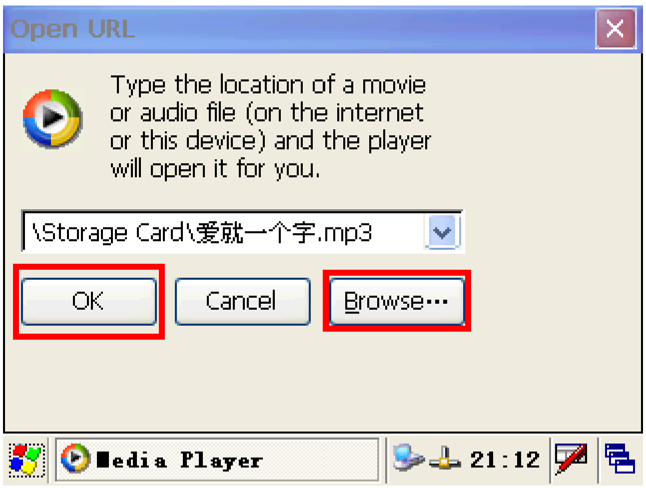
1) play MP3 music by Media Player

Double-click the icon “Media Player” on the desktop. Windows Media Player starts to run, as shown in the following diagram:

Click “File->Open”:



Locate the mp3 file:



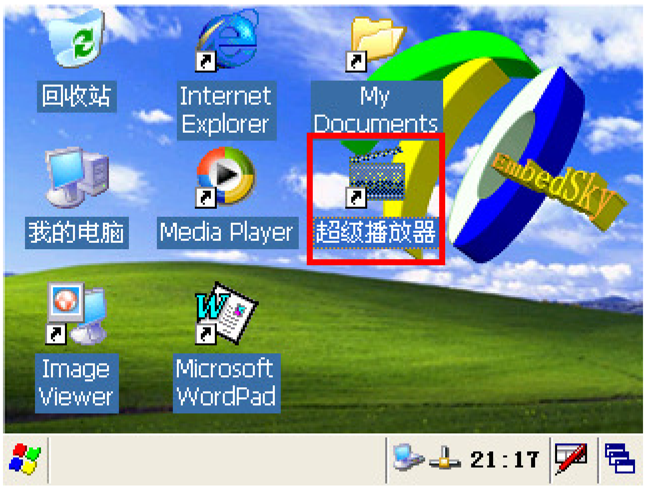
Start playing:

Media Player also supports the file of WMV format.

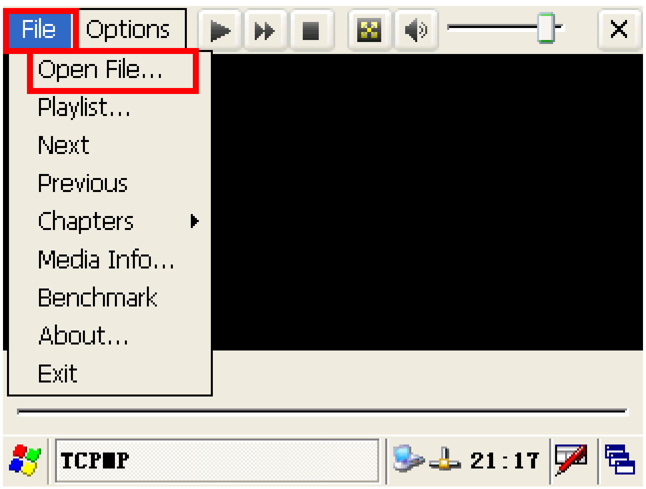
2) play MPEG4 movie by "超级播放器"

"超级播放器" is a commonly used player in Windows Mobile, similar to "暴风影音". This player supports the format of mpeg2, mov, avi and so on.

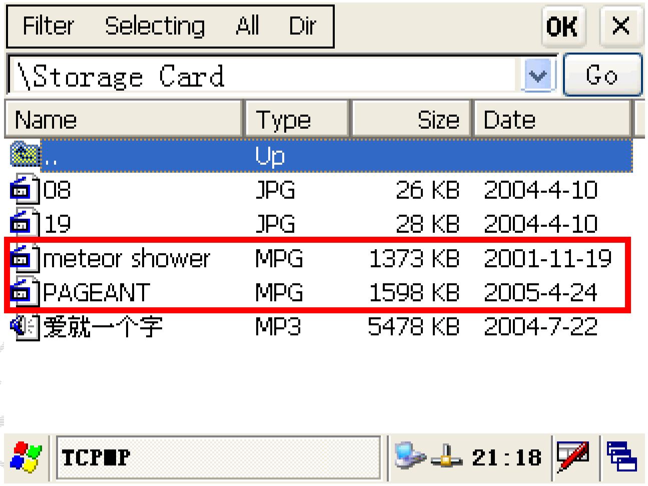
Double-click the icon "超级播放器" on the desktop:



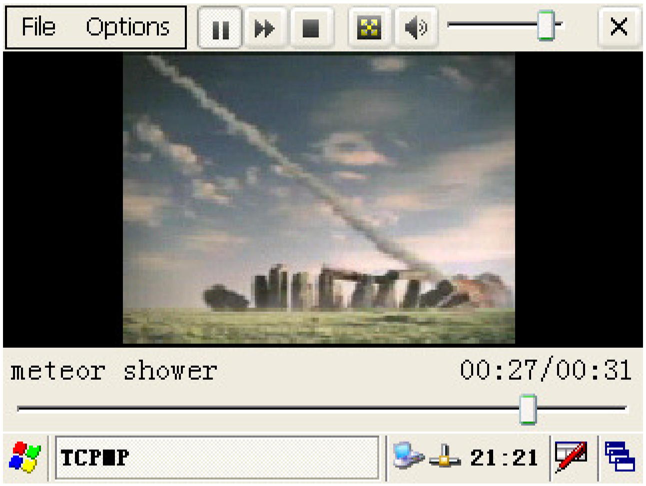
Click "File->Open File":



Select the movie file:



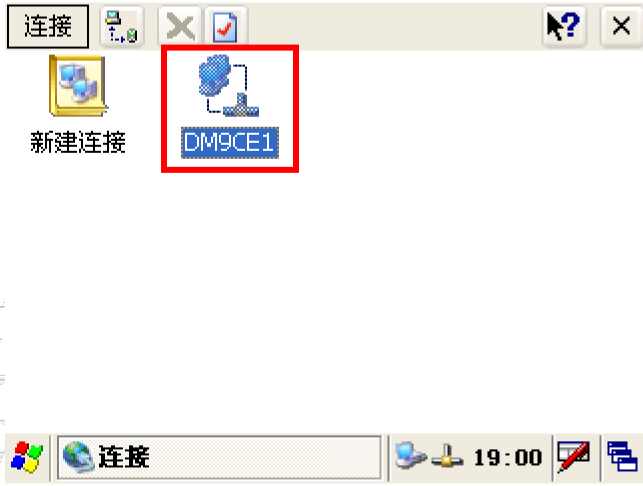
The screen capture image is shown in the following diagram:下图是播放中的截图：



3. 7. 5 Experiment of 100M network card test

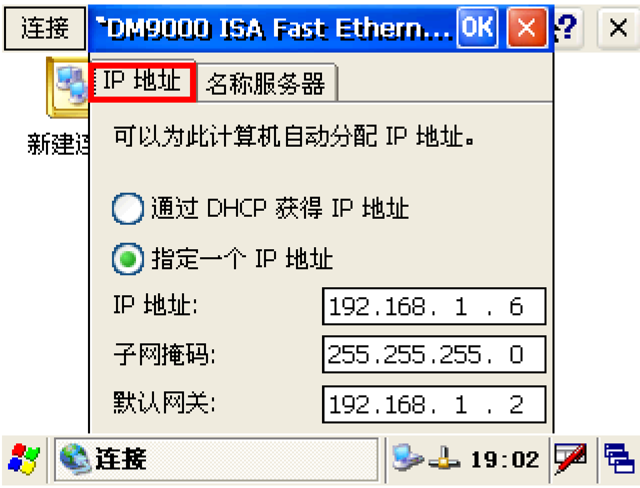
Click “开始->设置->网络和拨号连接”:

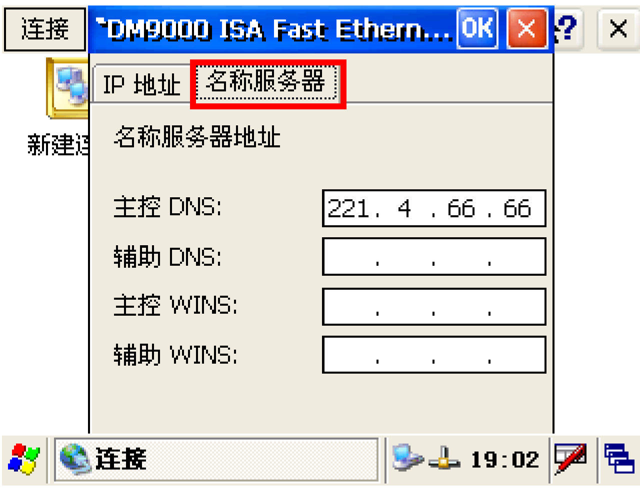
The following interface appears:



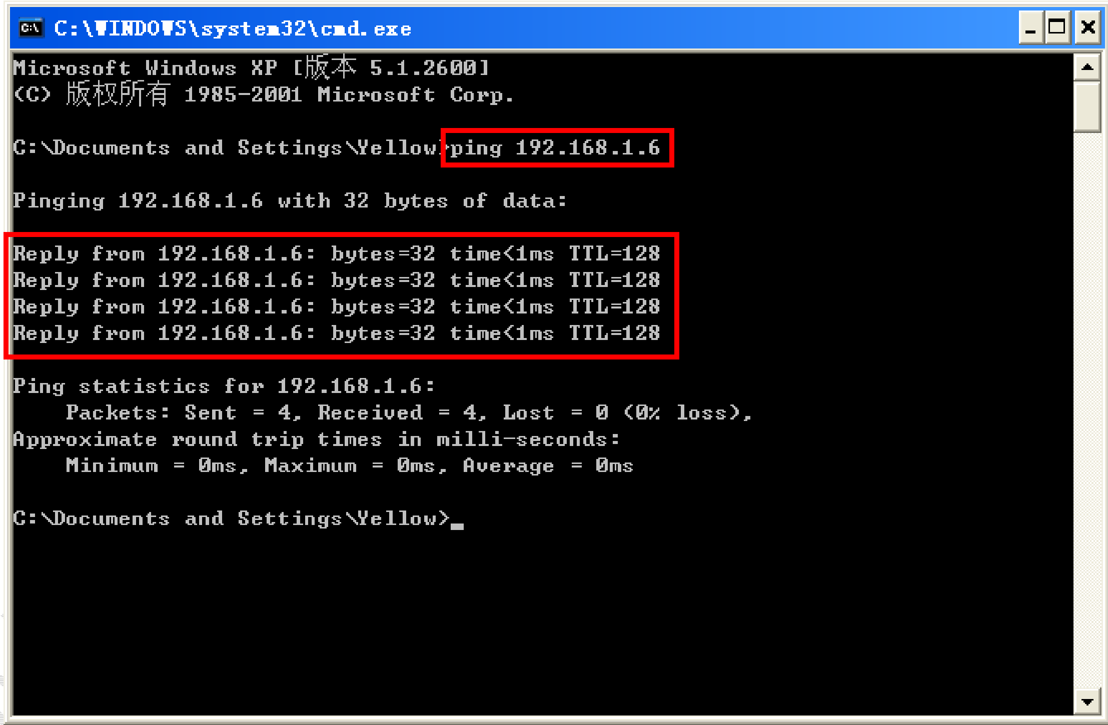
Double-click "DM9CE1" to open the configuration interface. We use the default parameters here as the following diagram shows. The user can set the parameters according to the actual situation:

The screen capture image of 100M network card is shown in the following diagram:





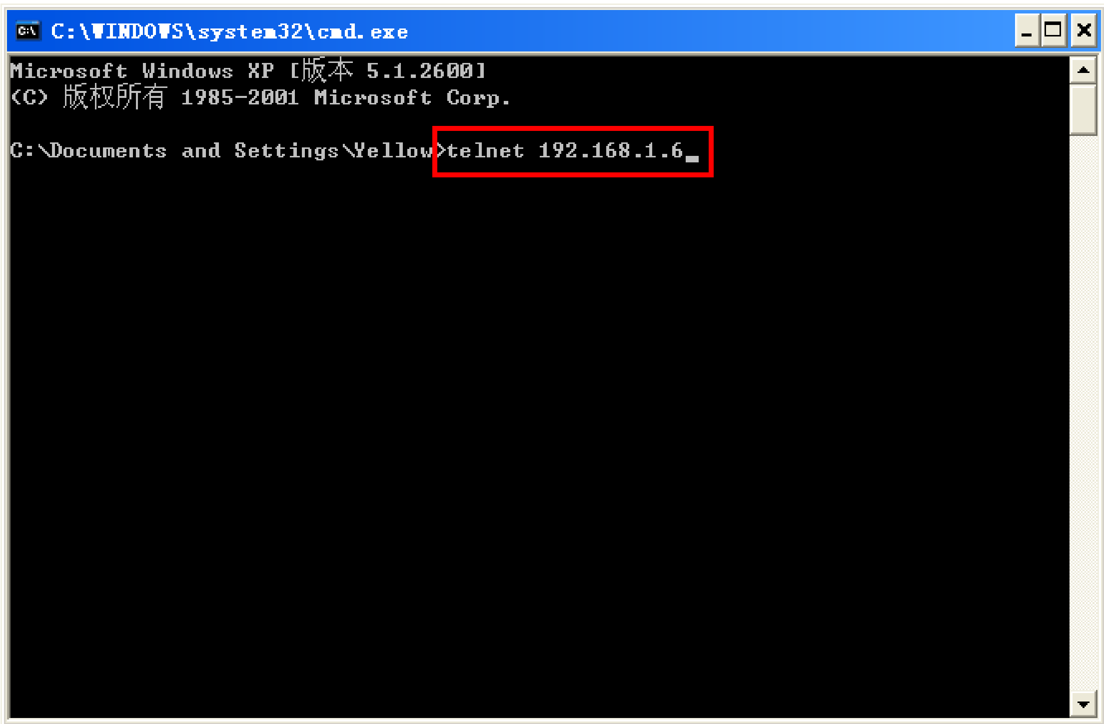
Execute the command "ping" on PC to test the connection state of network:



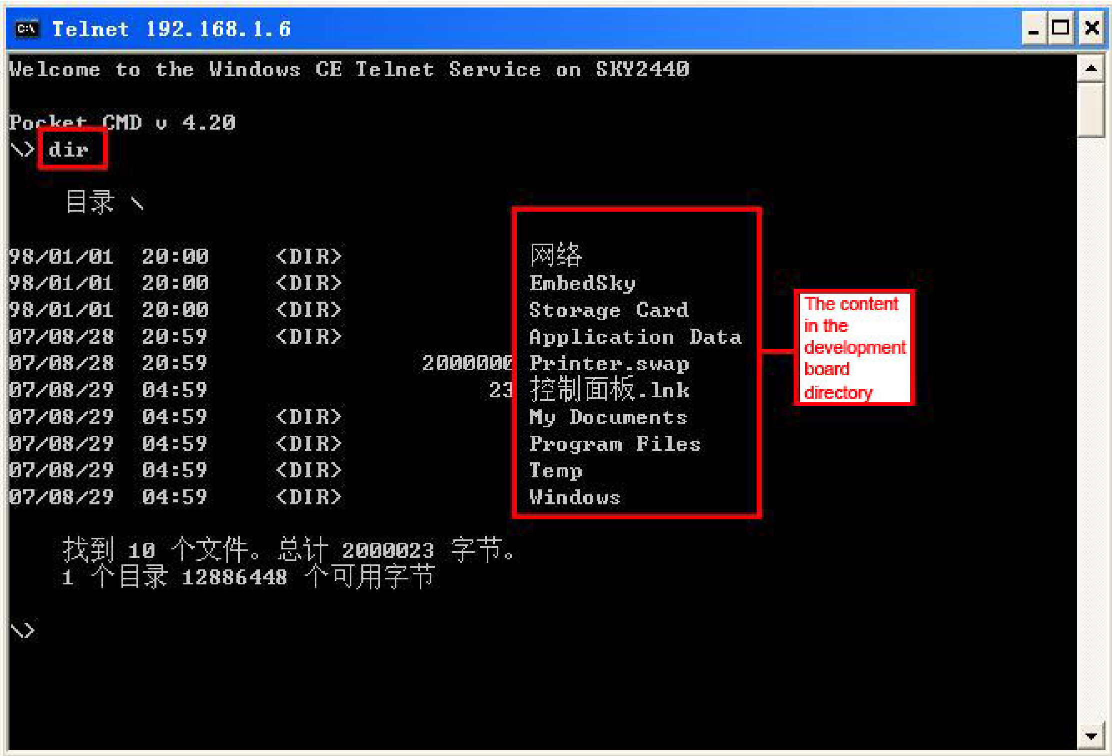
3. 7. 6 Experiment of telnet remote log-in

The telnet service runs automatically after WinCE start-up in TQ2440. Connect the net wire, and the user can log in platform remotely by telnet.

Enter the command “telnet 192.168.1.6” in command window:



Press the return-key to enter into platform console:

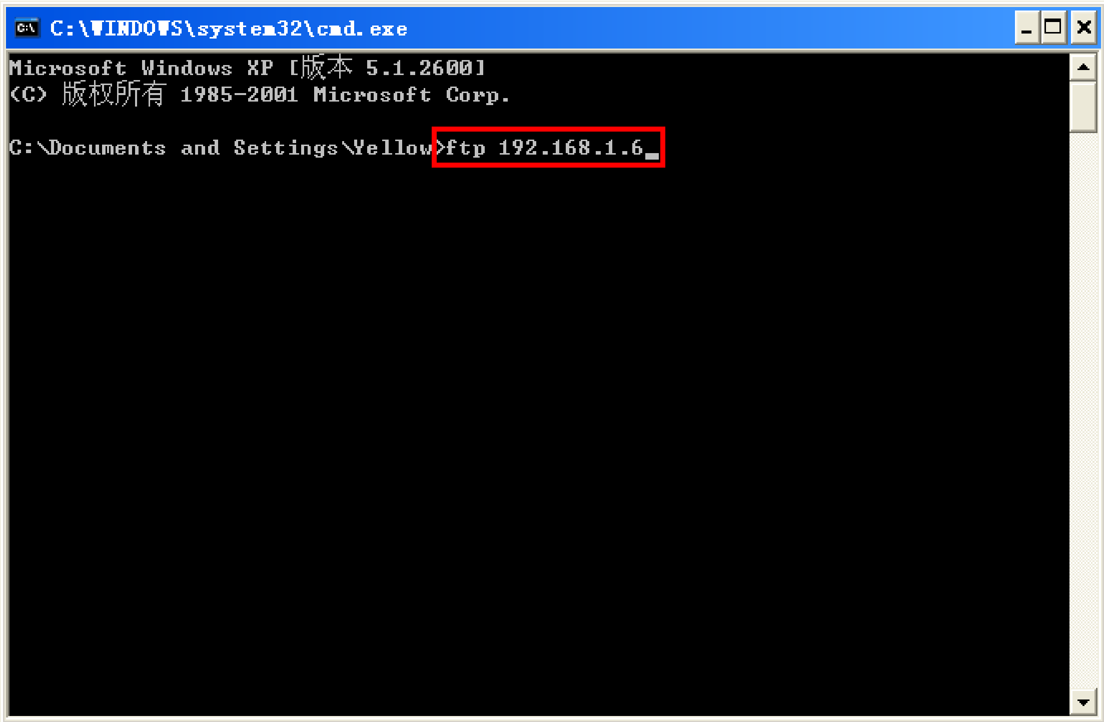


Caution: The default IP address in WinCE is “192.168.1.6”. No user name and password are needed when log-in.

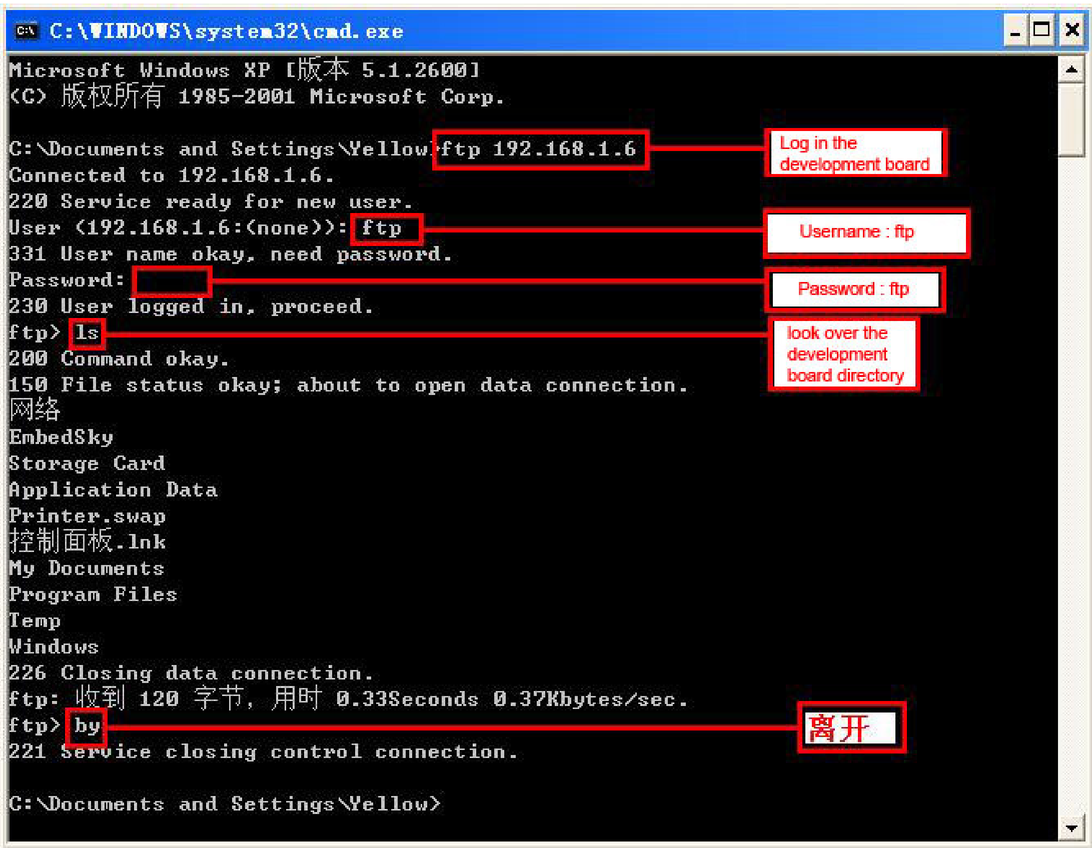
3. 7. 7 Experiment of FTP remote file transmission

The FTP service runs automatically after WinCE start-up in TQ2440. Connect the net wire, and the user can log in platform remotely by FTP.

Enter the command “ftp 192.168.1.6” in command window:



Press the return-key to log in platform. Input user name and password (ftp):

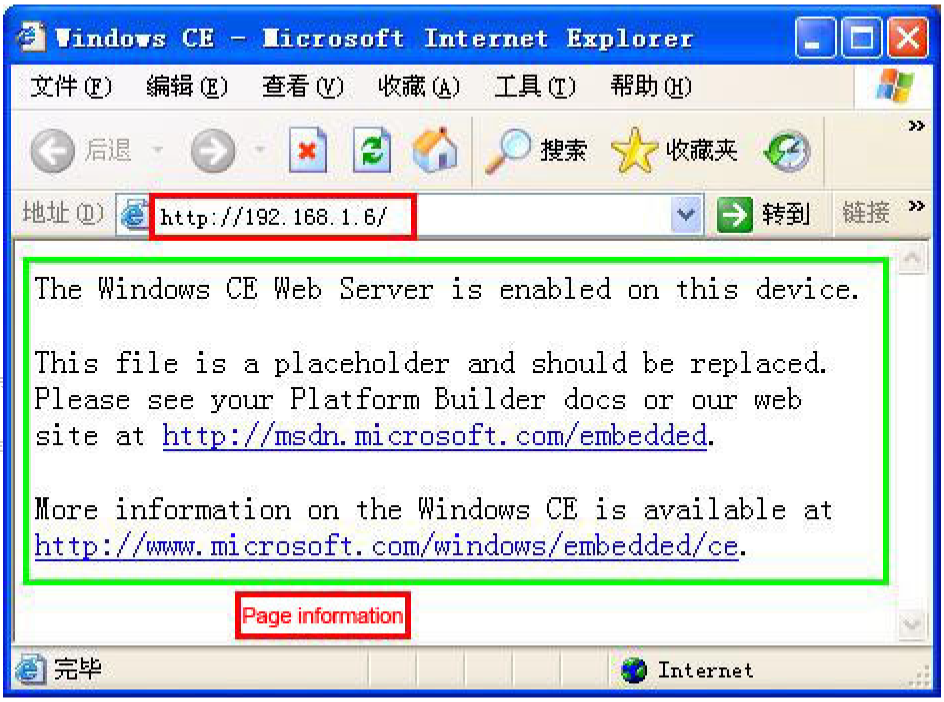


Caution: The default IP address in WinCE is "192.168.1.6". User name and password are both: ftp.

3. 7. 8 Experiment of Web server test

The http service, namely Web service runs automatically after WinCE start-up in TQ2440. Connect the net wire, and the user can access webpages provided by platform.

Enter “192.168.1.6” in address table and press return-key.

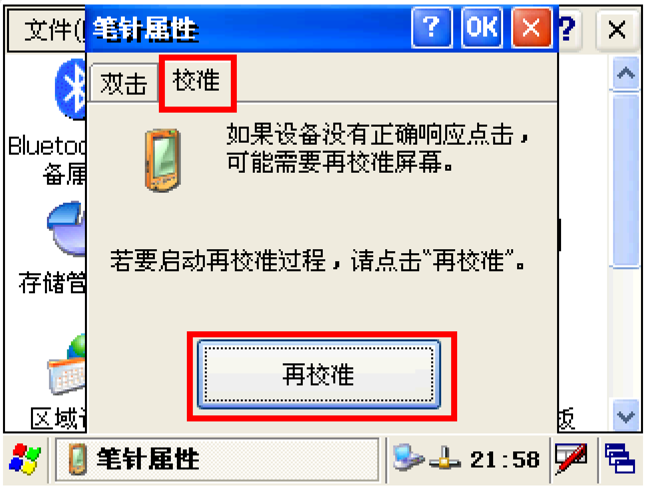


3. 7. 9 Experiment of touch-screen correction

The correction is needed when touch-screen doesn’t work correctly. Connect the USB mouse and click “开始->设置->控制面板”:

Find the icon “笔针” and double-click it:

The interface "笔针属性" appears. Click the tag "校准" to enter into correction interface:

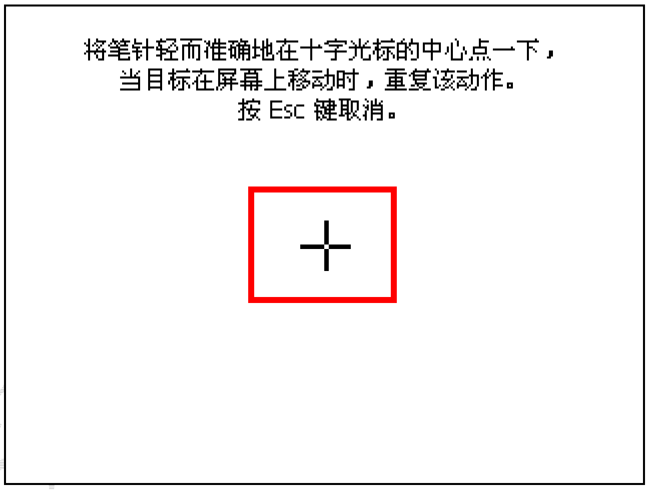


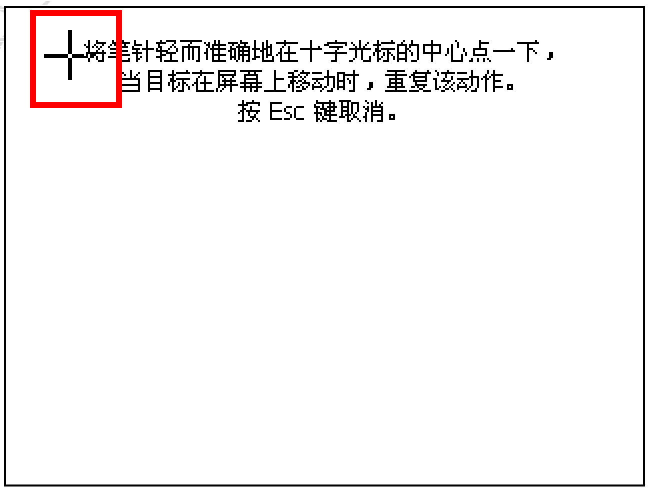
Click "再校准" to start correction:

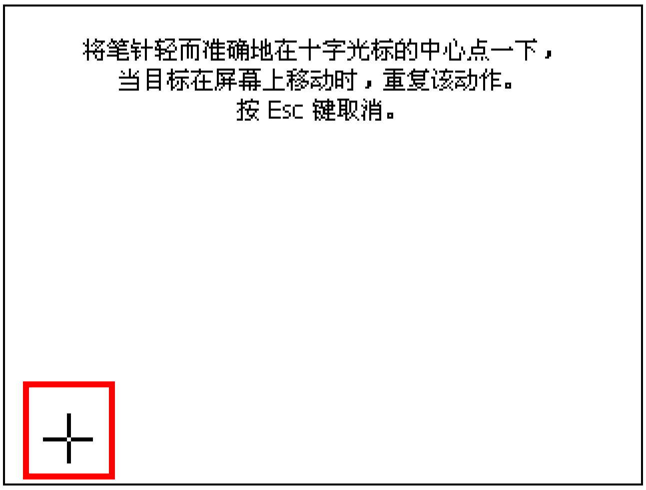
You could also choose to double-click the shortcut "校正触摸", as shown in the following diagram:

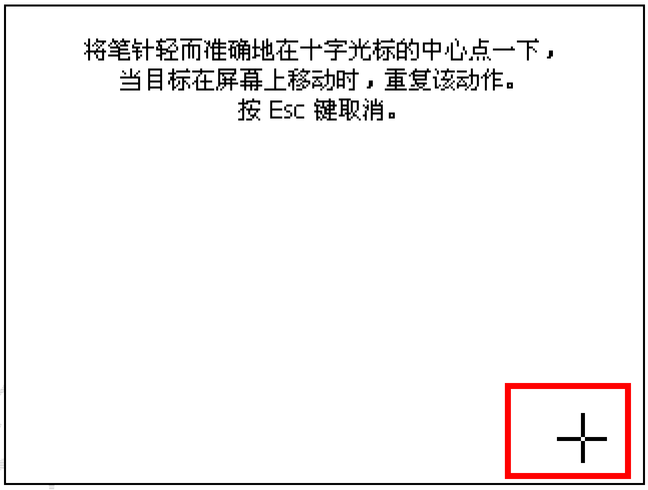
Correction starts:

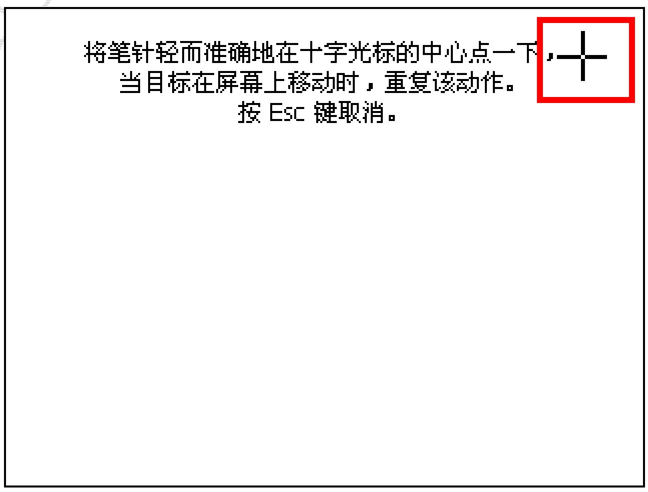
(caution: There are totally 5 points need to corrected. The center of the cross is the correction point. Touch the 5 correction points precisely for about 1 second each)


将笔针轻而准确地在十字光标的中心点一下，
当目标在屏幕上移动时，重复该动作。
按 Esc 键取消。

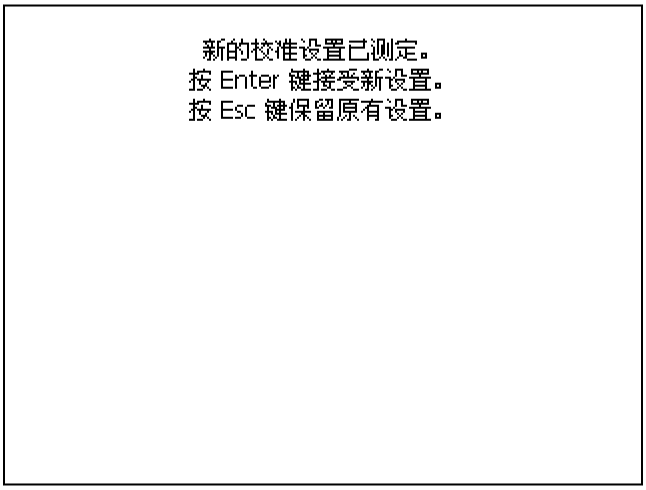

将笔针轻而准确地在十字光标的中心点一下，
当目标在屏幕上移动时，重复该动作。
按 Esc 键取消。


将笔针轻而准确地在十字光标的中心点一下，
当目标在屏幕上移动时，重复该动作。
按 Esc 键取消。

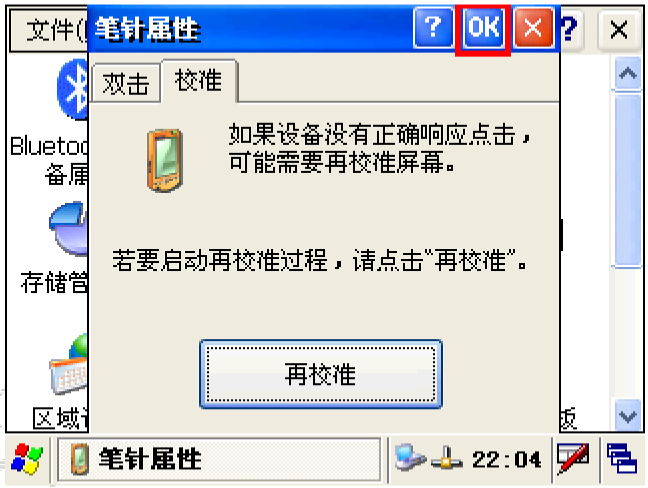




The following diagram appears after correction. Click any place of the touch-screen to continue:



Get back to the following interface. Click “OK” to finish the correction:

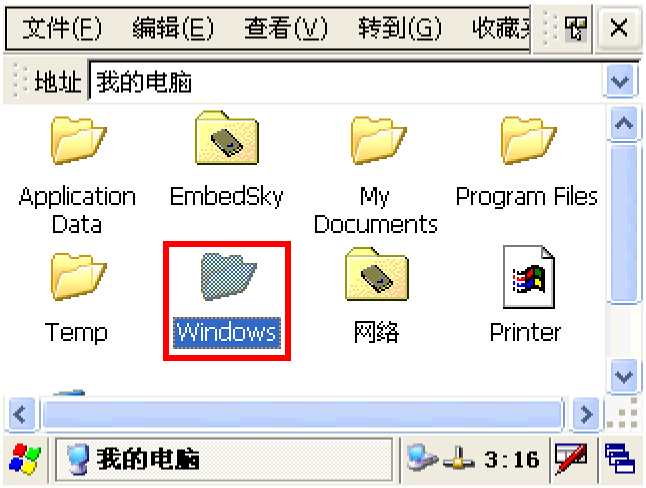


(caution: The touch-screen test value contained in BSP package is derived from plenty of actual tests. The user can also consult “section 4.8” to find a better value, and edit it into OS kernel. After the kernel re-compilation, there is no need to do correction any more.)

3. 7. 10 Experiment of USB camera test

The OS kernel contains USB camera driver which supports Z301P camera chip. Insert the USB camera into the USB interface on platform and start up the platform. After running the test application, the image captured by camera will appear on LED.

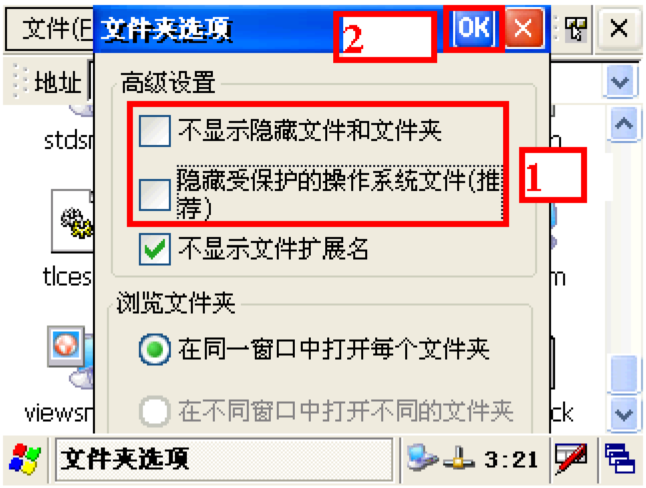
The test application has been compiled into OS kernel. It is under the hidden directory “Windows”.



Open the directory “Windows” in WinCE:

Select the menu “查看” and click “选项(O)”:

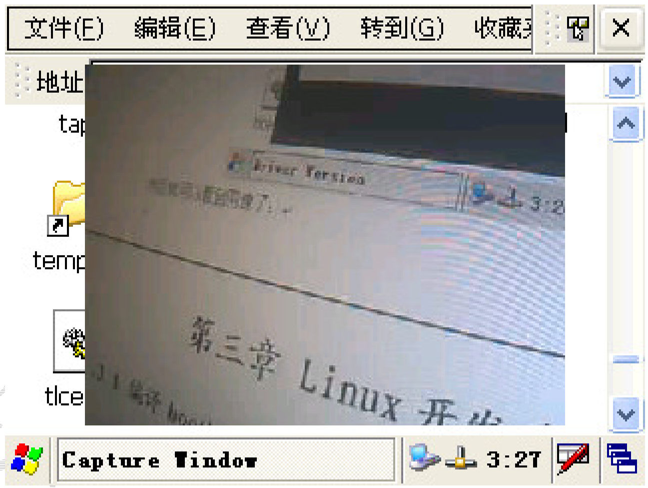
Deselect the options “不显示隐藏文件和文件夹” and “隐藏受保护的操作系统文件(推荐)” and click “OK” to continue:



The USB camera test application “Testzc030x” can be found under the directory “Windows”:

Double-click the test application. Click “OK” in the following 2 pop-up interfaces:

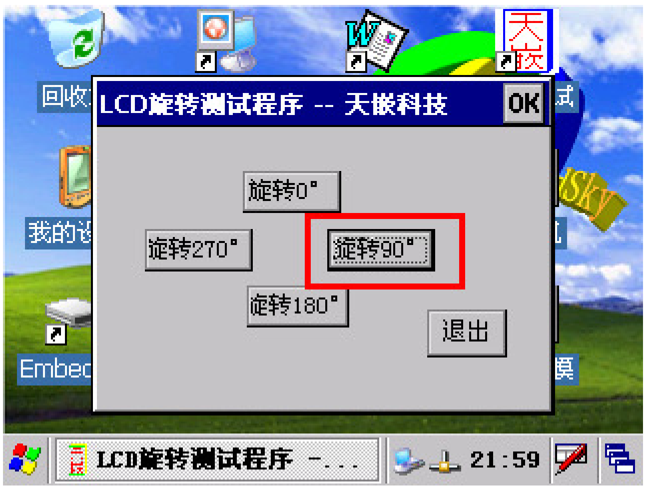
The captured image appears (the manual is in the image):



3. 7. 11 Experiment of image rotation in LCD

Click the shortcut “LCD 旋转测试” on the desktop, as shown in the following diagram:

The following interface appears:



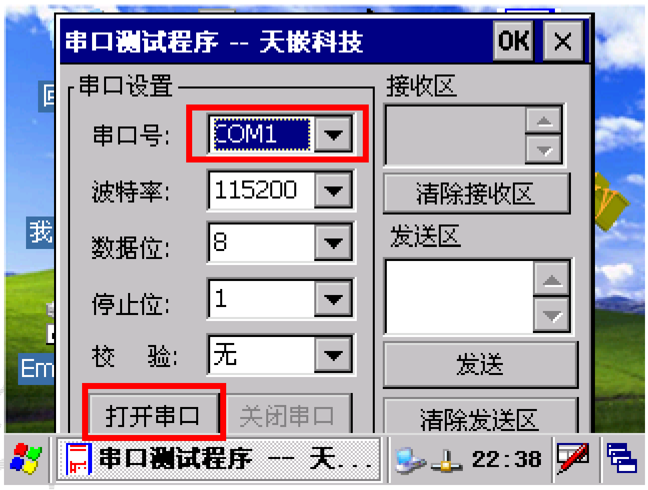
Click the option “旋转 90°”. The desktop image rotates by 90 degree, as shown in the following diagram:

270 degree rotation and 180 degree rotation are similar to the upper case.

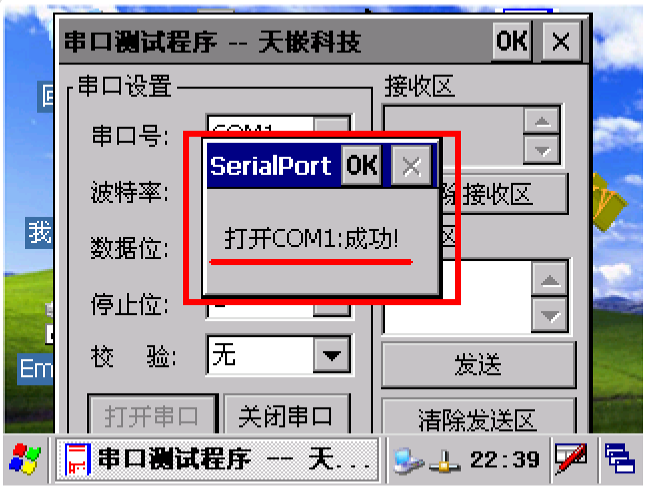
3. 7. 12 Experiment of 3 serial ports test

Click the shortcut “串口测试” on the desktop, as shown in the following diagram:

The serial port test application starts to run:



Select COM1 and click "打开串口". The prompt interface appears in the following diagram:

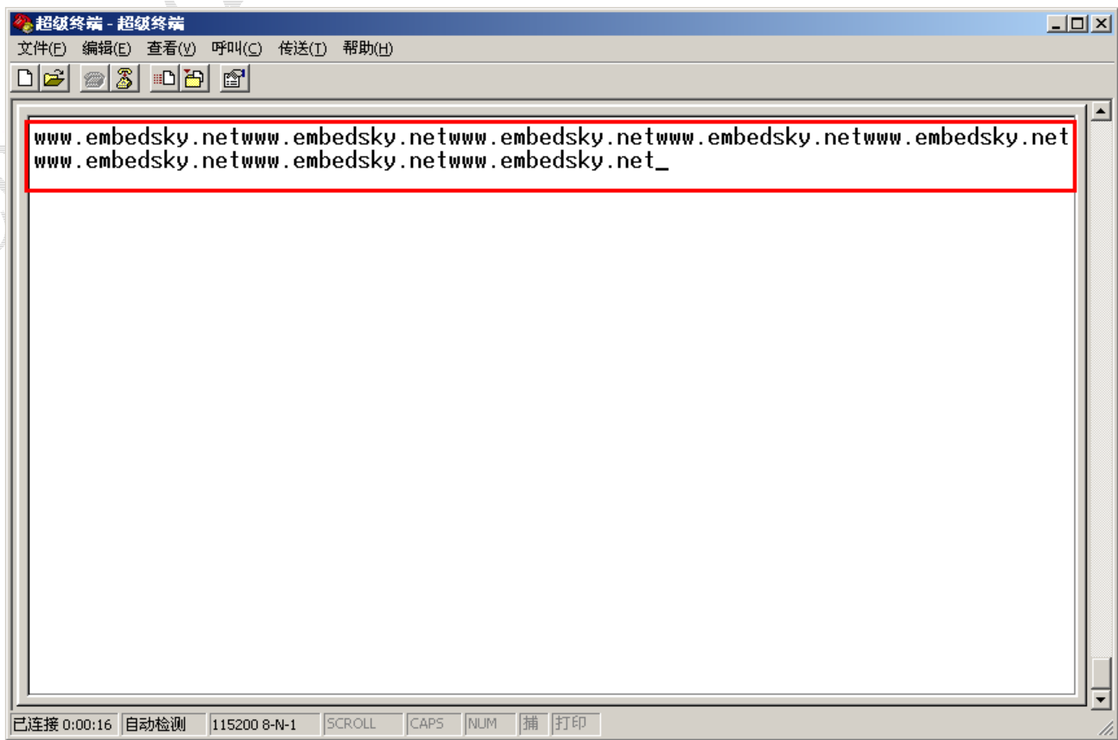


Use serial port tool in PC to send data to platform. We enter "TQ2440" in PC and send the string. Then we find that the platform has received the string:

The information entered in platform can also be send to PC. We enter "www.embedsky.net" in platform and send the string. The string is received by PC, as shown in the following diagram:

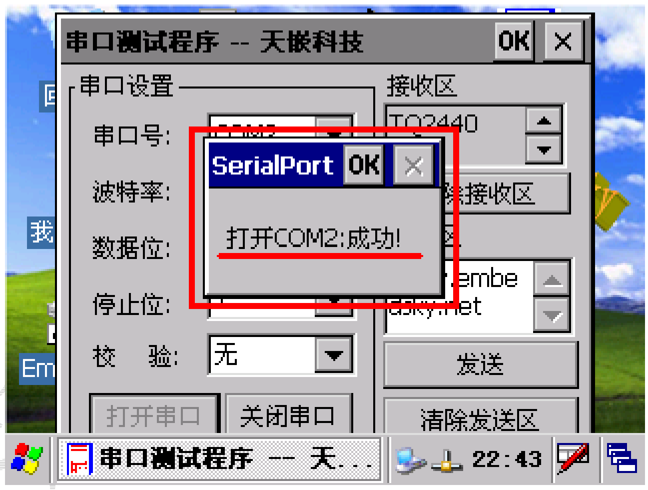
In the platform:

In PC (receive the data more than once):

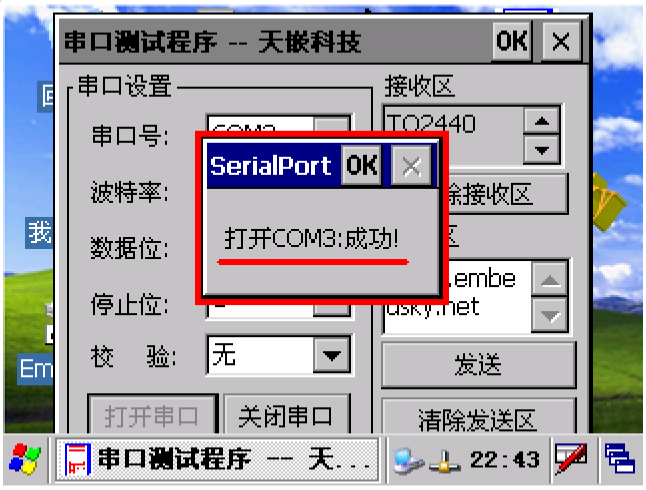


The process of testing COM2 and COM3 is similar to COM1 test. The following diagrams show that COM2 and COM3 are open successfully:

COM2 is open successfully:



COM3 is open successfully:



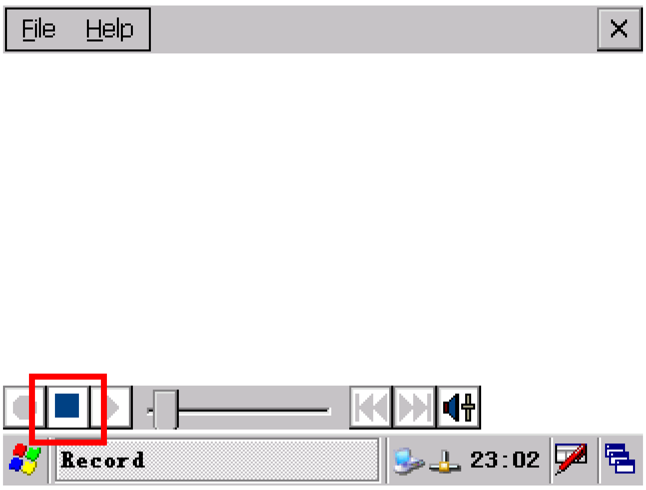
3. 7. 13 Experiment of recording test

Find the shortcut “录音机” on the desktop, as shown in the following diagram:

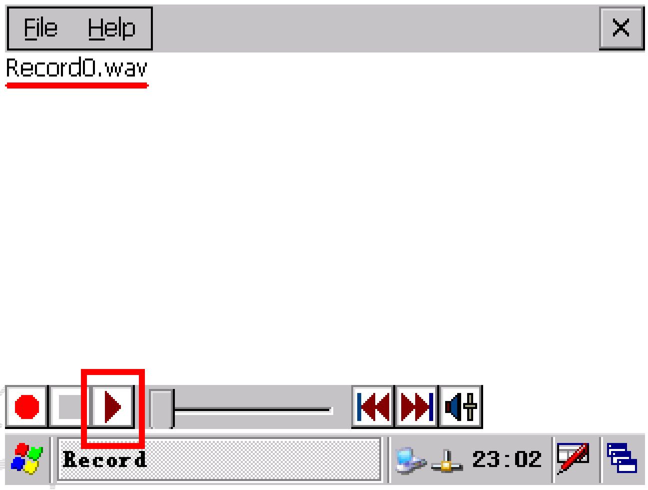
Double-click it to start recording:

Connect the earphone and microphone, and click the following red round button to start recording:

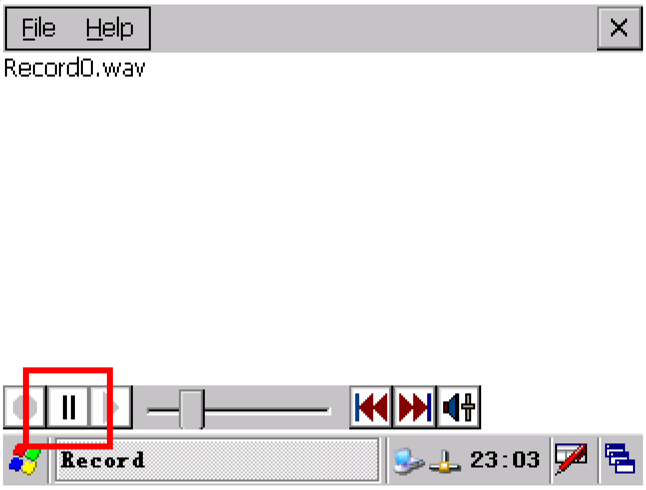
Click the blue square button to stop recording:



Click the brown triangle button to replay the recording:

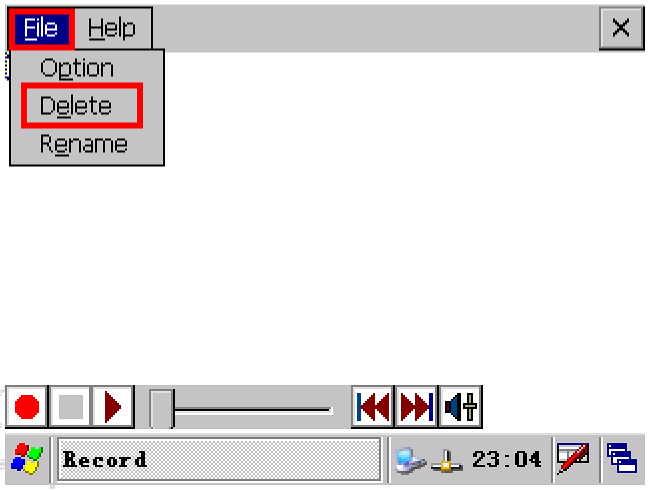


Click the double-vertical-line button to stop playing:

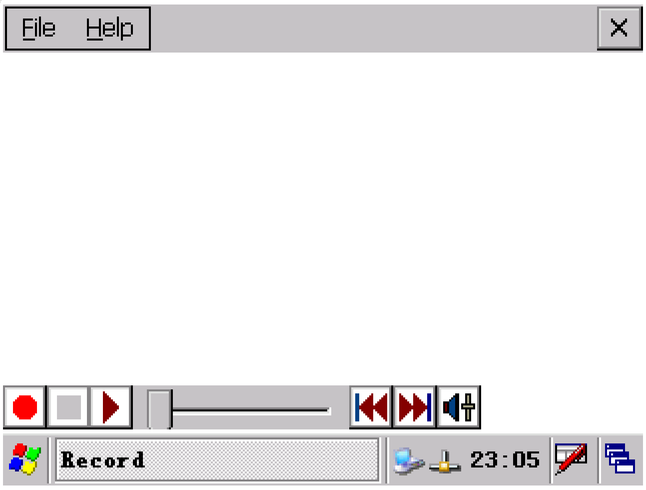


If the user wants to delete the recording file, select the file first:

Click “File->Delete” to delete the file:

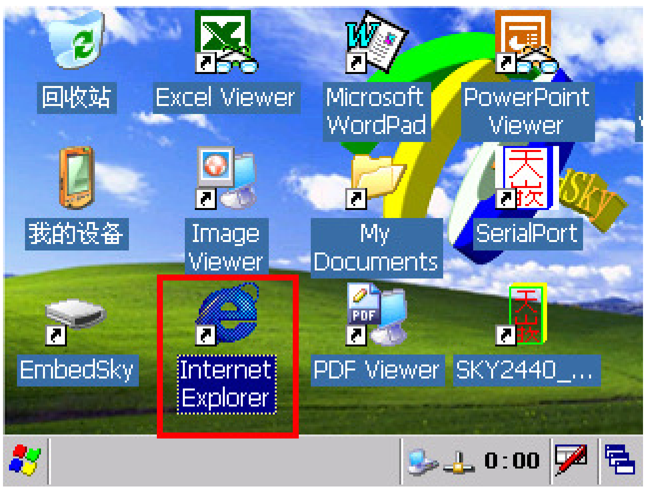


The recording file has been deleted:

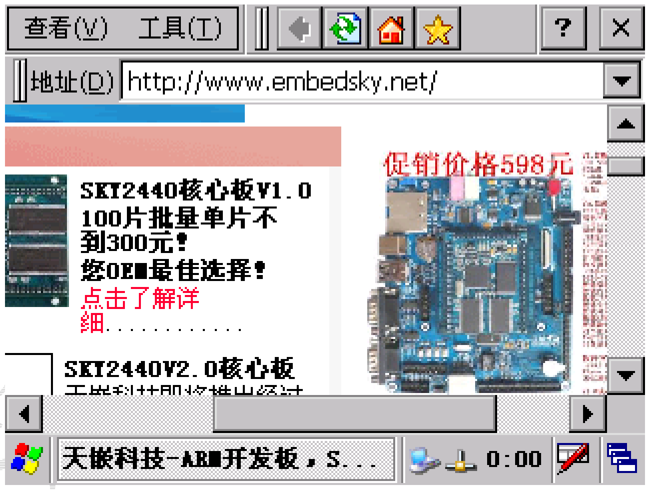


3. 7. 14 Experiment of surfing by IE explorer

Find the shortcut “Internet Explorer” on the desktop:



Start the IE explorer and open the website www.embedsky.net:



The capture image of “天嵌科技” forum:

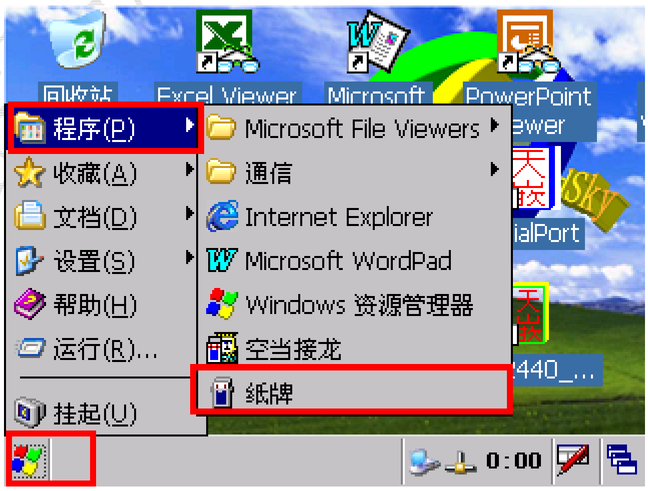
The capture image of WinCE plate:

3. 7. 15 Experiment of USB synchronization by ActiveSync

Please consult “5.5 节”.

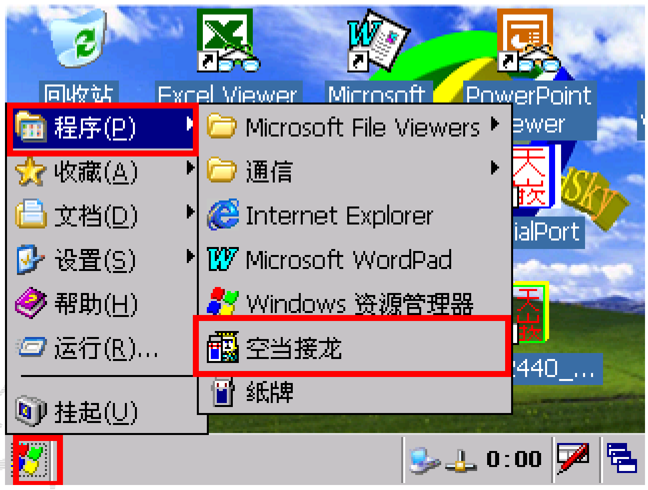
3. 7. 16 Experiment of WinCE self-carried game test

Click “开始->程序->纸牌”:

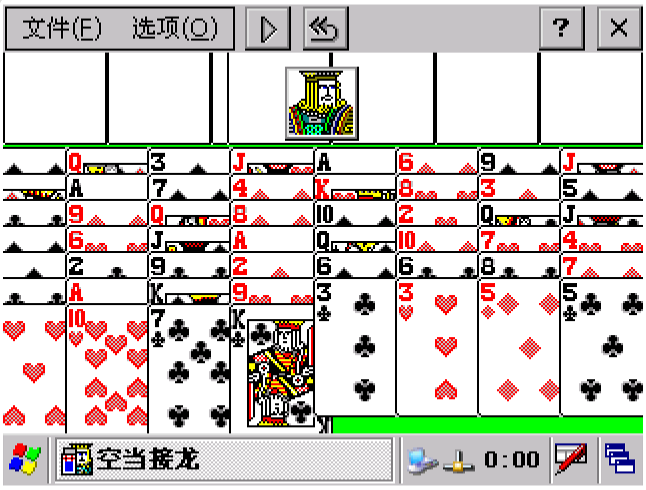


The playing card game:

Click “开始->程序->当空接龙”:

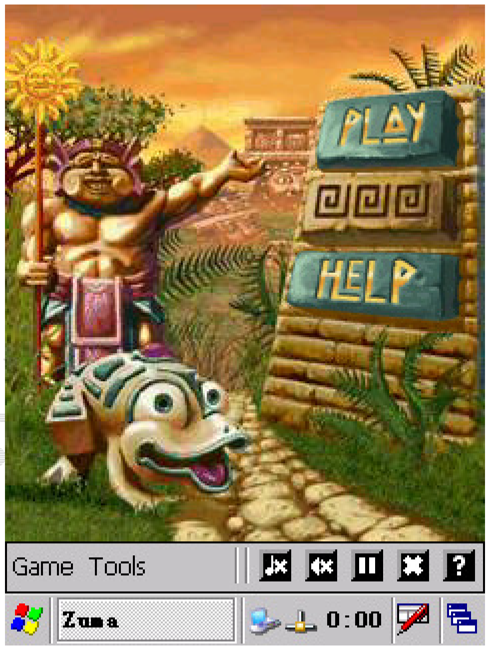


Start the game:

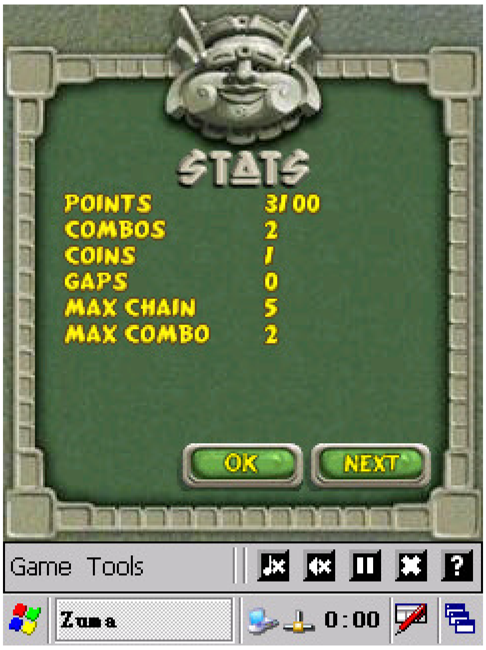


3. 7. 17 Experiment of Zuma game test

The running interface of Zuma game:


PLAY
HELP
Game Tools
Zuma
0:00

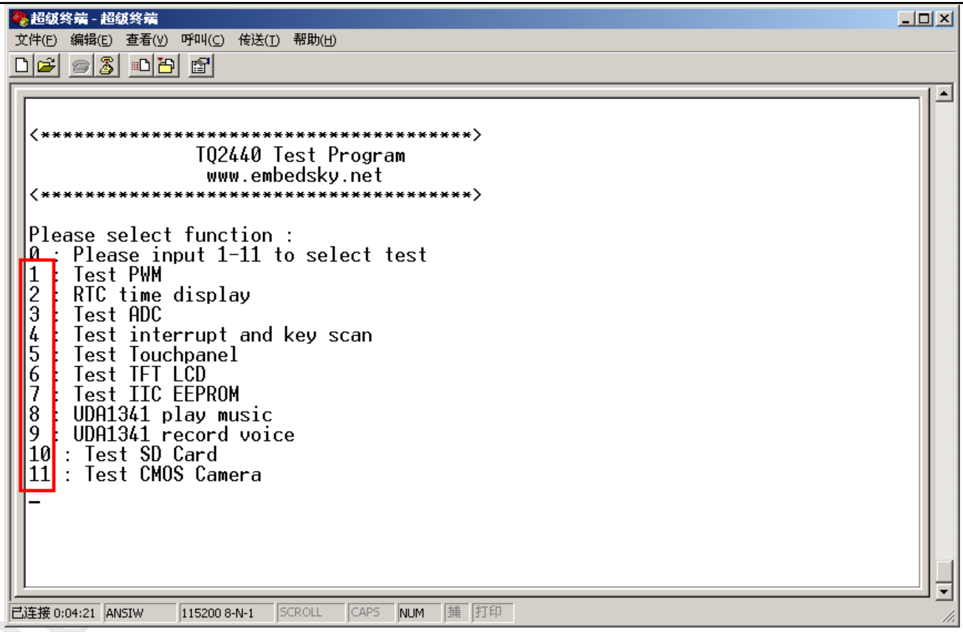

STAGE1
BACK
TEMPLE OF ZUKULKAN
PLAY
Game Tools
Zuma
0:00



3. 8 Experiment of non-OS testing Demo

Download the non-OS testing Demo to SDRAM, or burn it to Nor Flash or Nand Flash. Please consult "3.3 节"(use H-Jtag to burn Nor Flash), or "Step7" of "3.4.3 节"(download the application to SDRAM), or "Step11" of "3.4.3 节"(burn application to Nor Flash).

Execute the non-OS testing Demo:



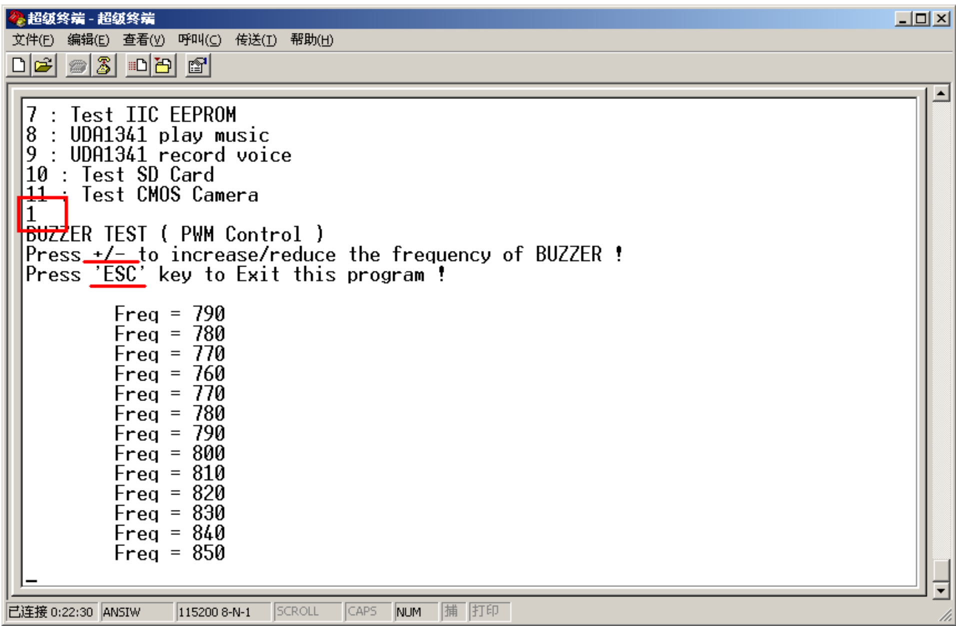
Enter the upper number according to prompt to start testing:

Caution: it is not suggested to select the option 10 (Test SD Card). Because when SD Card is being tested, a group of data would be written to SD card which probably destroy the original data. If the user tries to use the SD Card in PC or other device after doing this test, please format the SD Card first.

3. 8. 1 Experiment of PWM function test

This experiment needs a buzzer. The strength of sound made by buzzer indicates the output power of PWM.

Use the key "+" and "-" on keyboard to increase and decrease the number of PWM pulses. Or press the key "ESC" to exit:

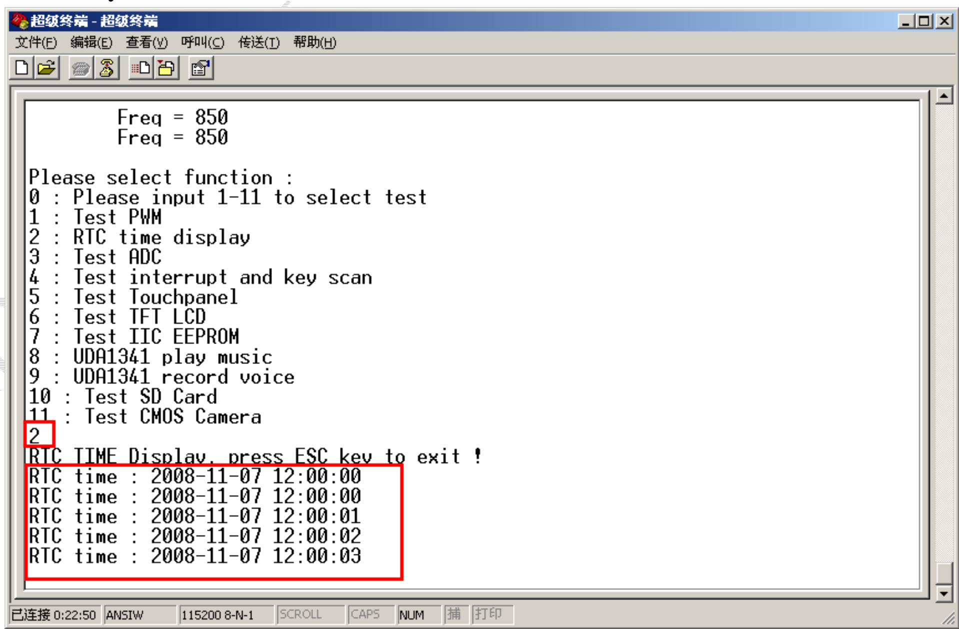


3. 8. 2 Experiment of real-time clock test

Write the test time to the relevant register of real-time clock, and then CPU will read it and print it out by serial port.

(be cautious to use this test. Because the test can probably modify the register of real-time clock. Thus you need to reset the real-time clock everytime when system starts up.)

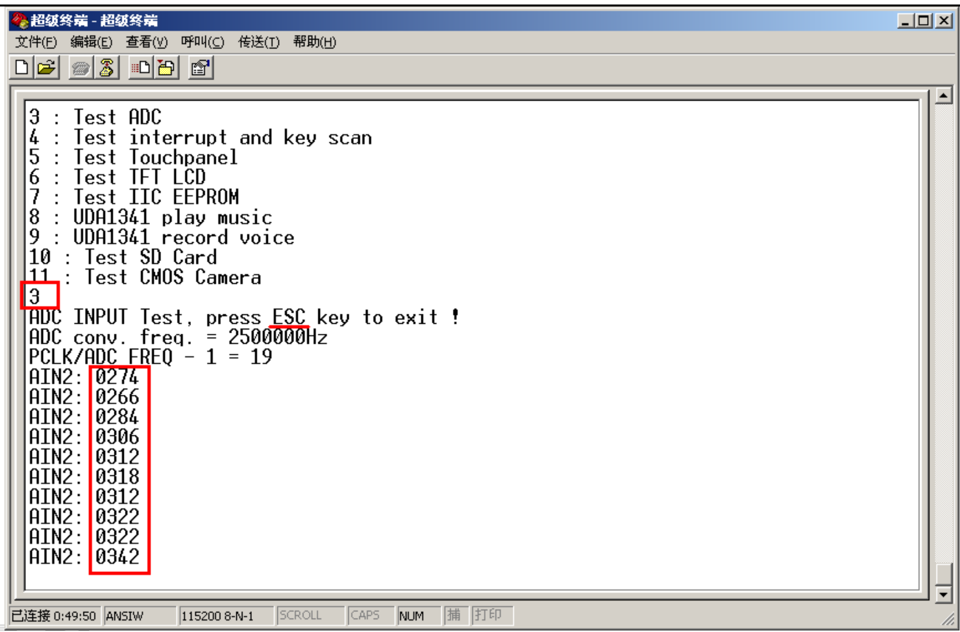
Press “ESC” on keyboard to exit the test:



3. 8. 3 Experiment of ADC conversion test

Rotate the screw of adjustable resistance can correspondingly change the strength of output from serial port. (caution: The ripple exists in reference voltage. Therefore, there will be tiny difference between print-out values even if the value of resistance doesn’t change.)

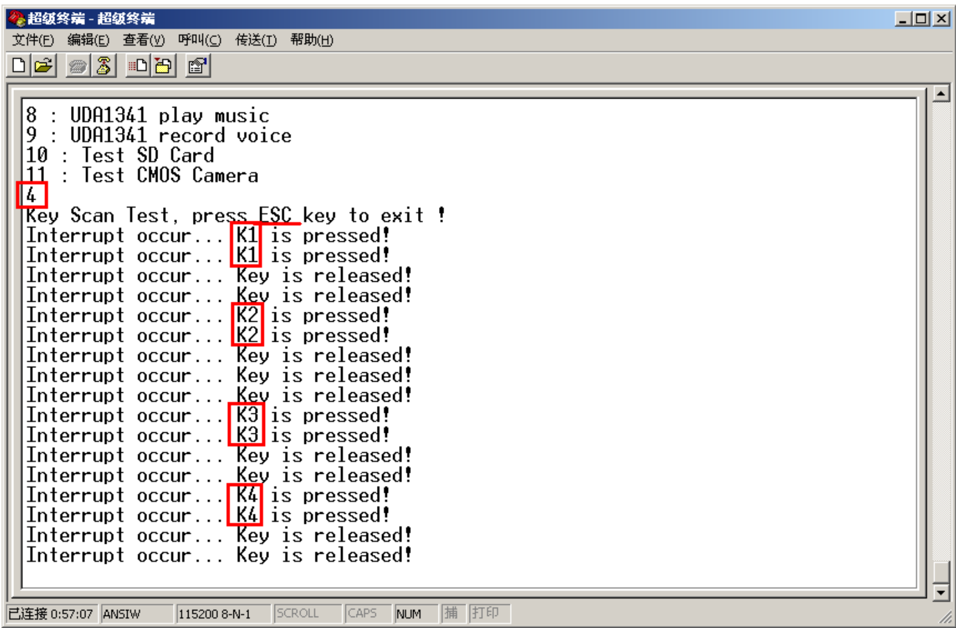
Press “ESC” on keyboard to exit:



3. 8. 4 Experiment of external interrupt test

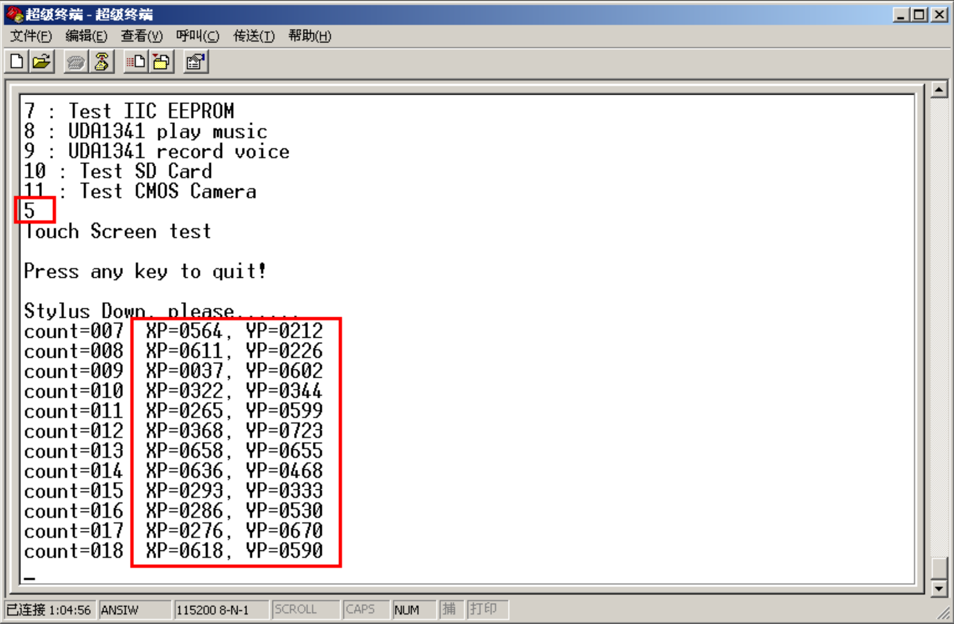
The user can press external button on TQ2440 to create interrupt signal. The corresponding information appears in the following diagram:

Press "ESC" on keyboard to exit:



3. 8. 5 Experiment of touch-screen test

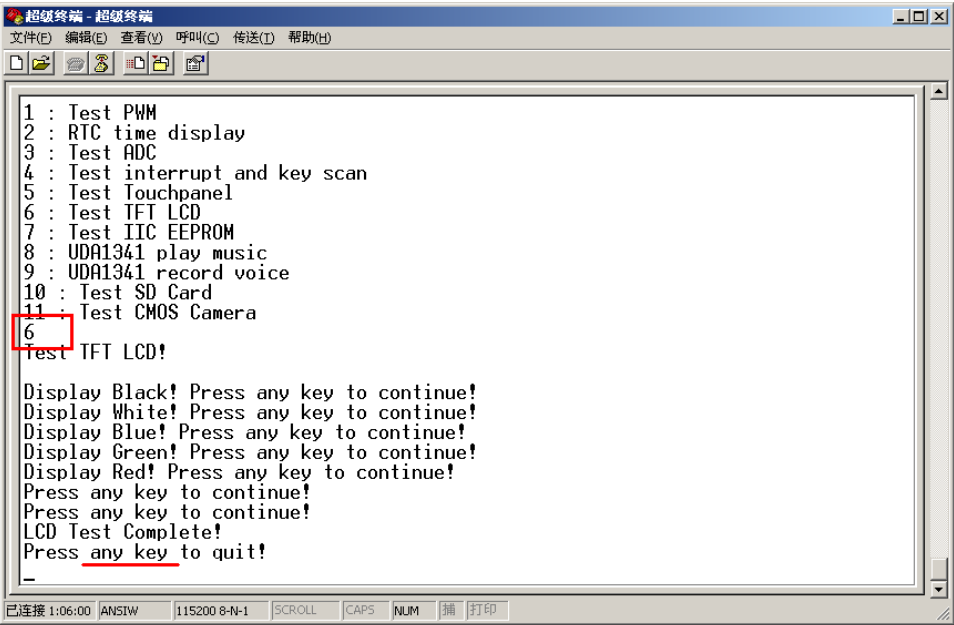
This test is similar to "3: Test ADC". Touch the touch-screen by using a pen to get the test values. Press any key on the keyboard to exit:



3. 8. 6 Experiment of LCD test

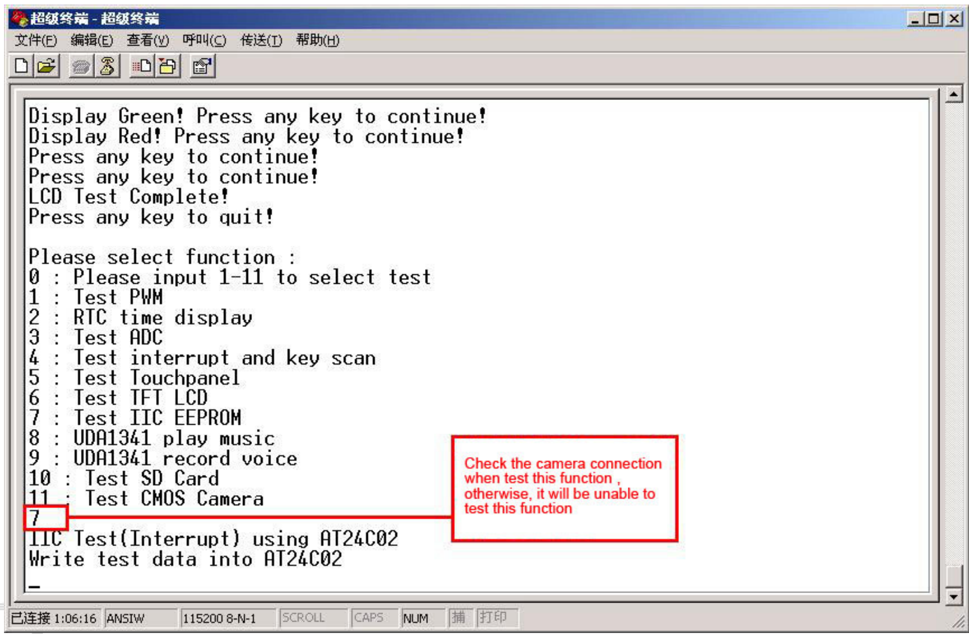
Press any key on the keyboard to start LCD test. The white color, blue color, green color, multi-color, flower and “EmbedSky” logo are displayed in sequence. Press any key to exit. The “EmbedSky” logo still appears on LCD after exit.

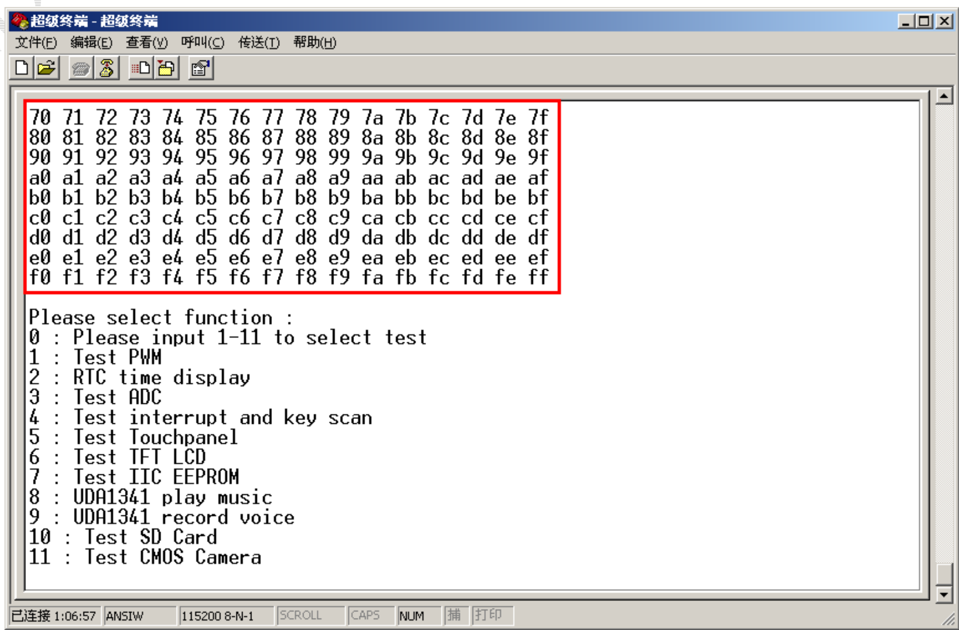
Press “ESC” on the keyboard to exit the test:



3. 8. 7 Experiment of IIC interface test

Use the chip AT24C02 to make the test. Write some data into the chip and then read it. Make sure to remove the camera before the test, otherwise the test may fail:

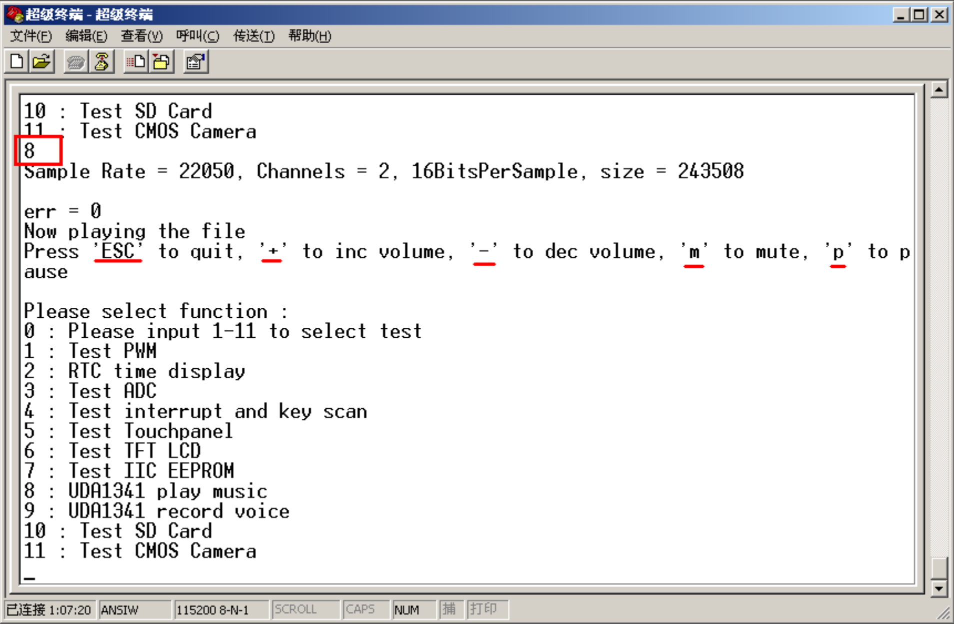




3. 8. 8 Experiment of audio output test

Insert the earphone to hear the start-up music of Windows.

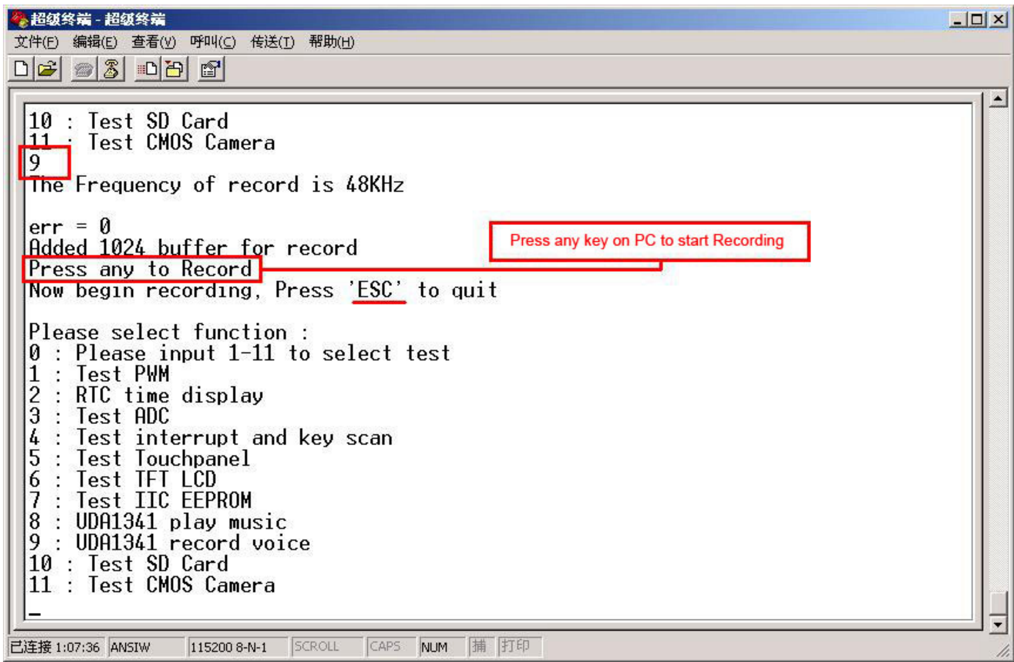
Press “ESC” to exit, “+/-” to increase and decrease the volume, “p” to pause and continue and “m” to mute:



3. 8. 9 Experiment of audio input test

Insert microphone and earphone. You can hear the voice from earphone when speak.

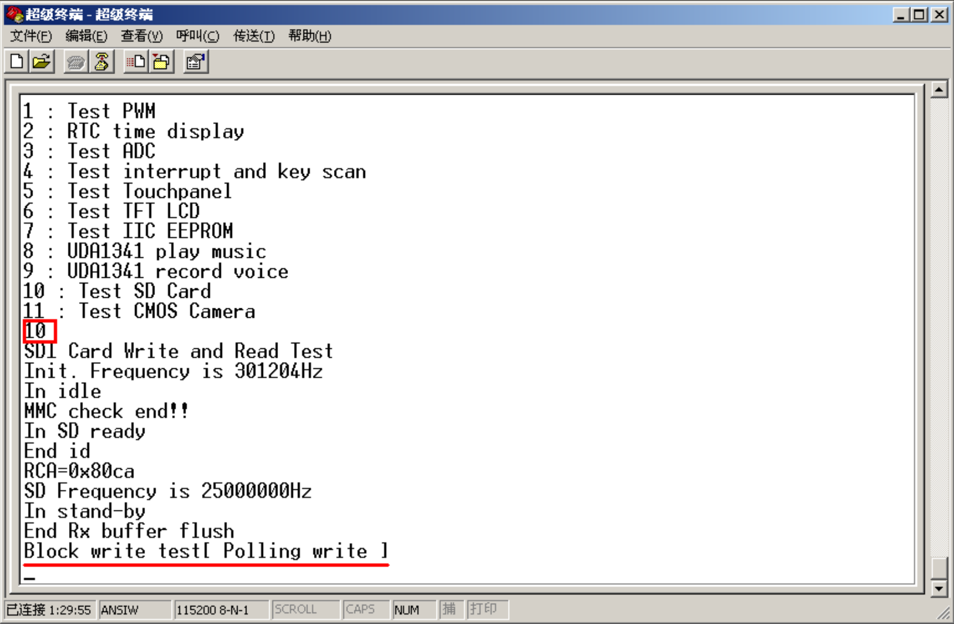
Press “ESC” on the keyboard to exit:

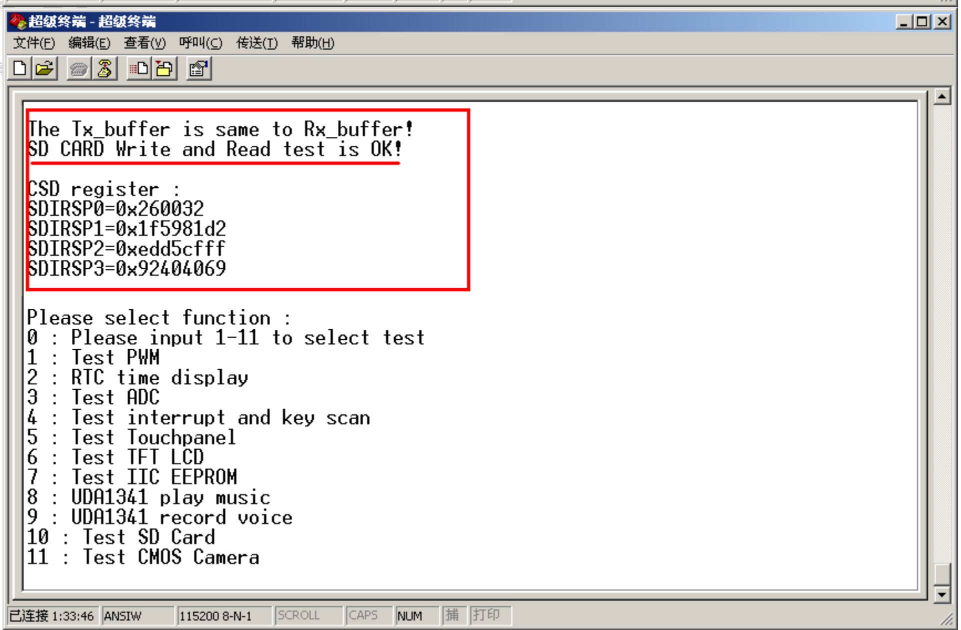


3. 8. 10 Experiment of SD card test

Write some data to SD card and then read it:

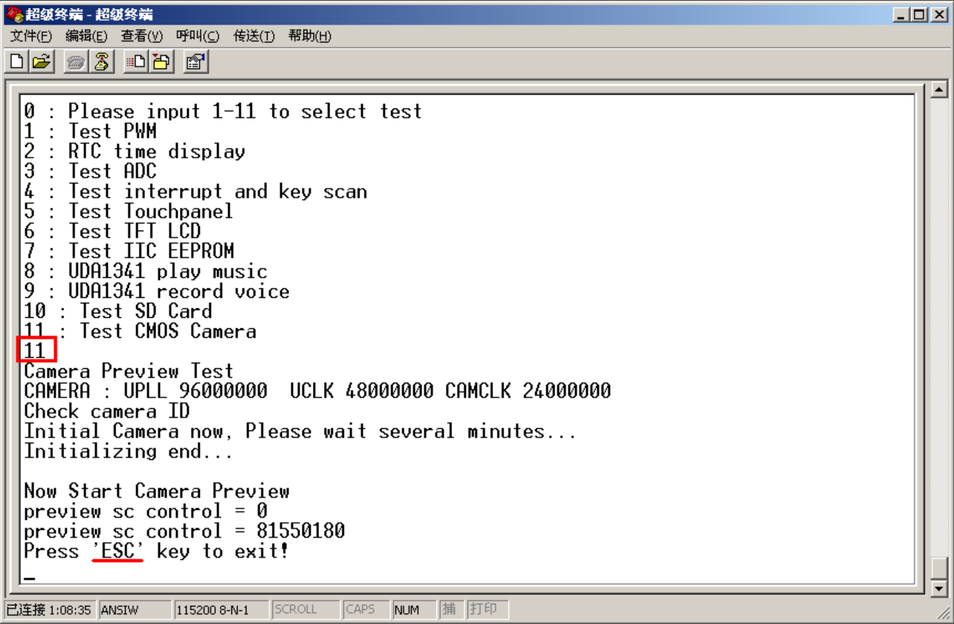
(be cautious to make this test. Some data would be written to the SD card during the test which can break the file system. After the test, you need to format the SD card in Windows OS for the further usage.)





3. 8. 11 Experiment of camera test

Connect the camera and initialize it. Then the images captured by the camera will appear on LCD. Press “ESC” on the keyboard to exit:


超级终端 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
0 : Please input 1-11 to select test
1 : Test PWM
2 : RTC time display
3 : Test ADC
4 : Test interrupt and key scan
5 : Test Touchpanel
6 : Test TFT LCD
7 : Test IIC EEPROM
8 : UDA1341 play music
9 : UDA1341 record voice
10 : Test SD Card
11 : Test CMOS Camera
11
Camera Preview Test
CAMERA : UPLL 96000000 UCLK 48000000 CAMCLK 24000000
Check camera ID
Initial Camera now, Please wait several minutes...
Initializing end...
Now Start Camera Preview
preview sc control = 0
preview sc control = 81550180
Press 'ESC' key to exit!
已连接 1:08:35 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

Chapter 4 Linux Development Manual

4. 1 Compiling bootloader

Caution: The cross-compiler of version 2.95.3 is needed when compiling bootloader. Make sure to use the correct version before the compiling:

```
root@HJ:/opt/EmbedSky
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@HJ EmbedSky]# /usr/local/arm/2.95.3/bin/arm-linux-gcc -v
Reading specs from /usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3/specs
gcc version 2.95.3 20010315 (release)
[root@HJ EmbedSky]#
```

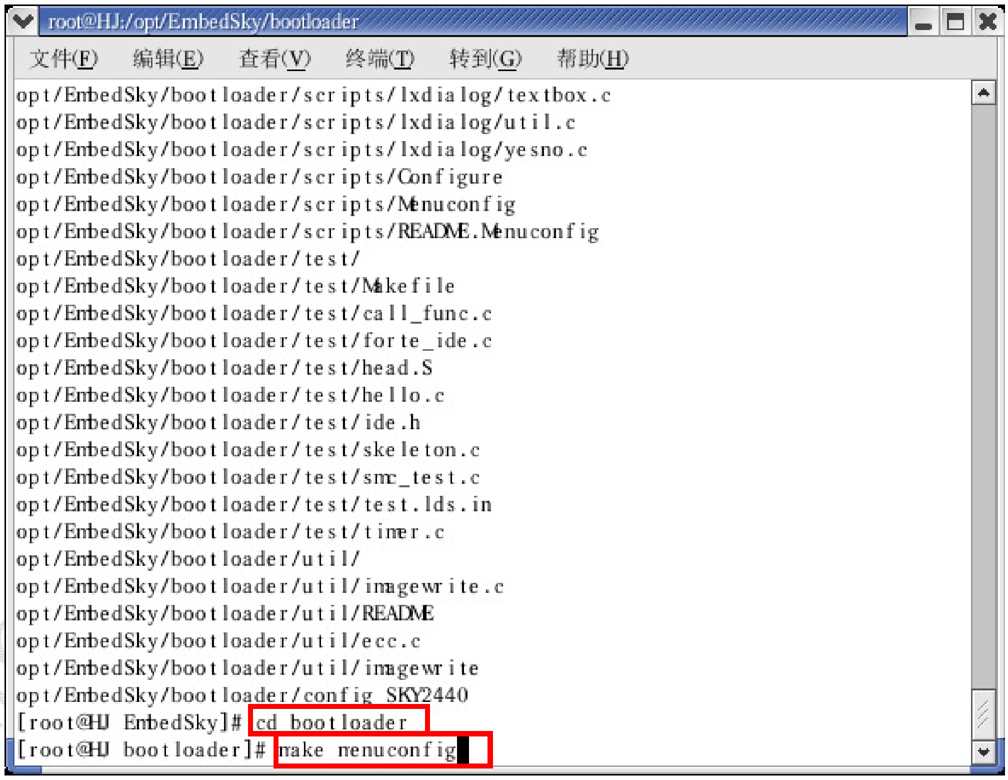
If the compiler version 2.95.3 has been successfully installed, the upper information above the red line will appear; Otherwise please consult “2.5.1” to install the compiler.

bootloader source code is under the directory “Linux/bootloader.tar.bz2/” in CD-ROM. Copy “bootloader.tar.bz2” to the directory “/opt/EmbedSky/” and decompress it:

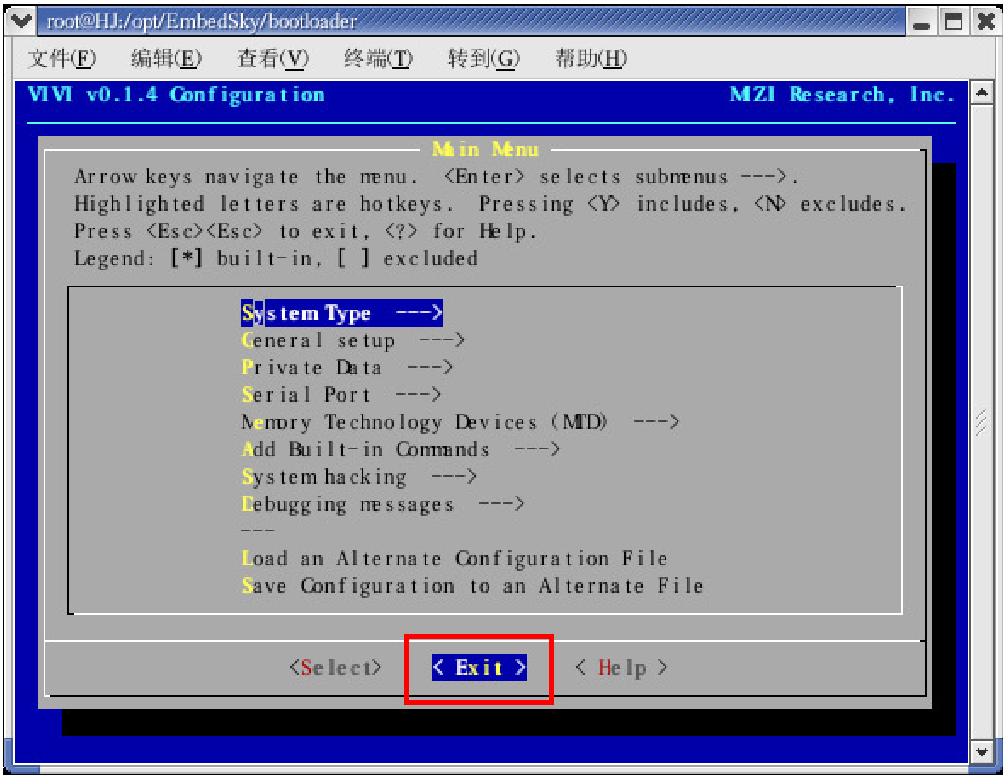
```
root@HJ:/opt/EmbedSky
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
[root@HJ EmbedSky]# tar xvfj bootloader.tar.bz2 -C /
opt/EmbedSky/bootloader/
opt/EmbedSky/bootloader/arch/
opt/EmbedSky/bootloader/arch/def-configs/
opt/EmbedSky/bootloader/arch/def-configs/smdk2440
opt/EmbedSky/bootloader/arch/Makefile
opt/EmbedSky/bootloader/arch/config.in
opt/EmbedSky/bootloader/arch/vivi.lds.in
opt/EmbedSky/bootloader/arch/s3c2440/
opt/EmbedSky/bootloader/arch/s3c2440/Makefile
opt/EmbedSky/bootloader/arch/s3c2440/head.S
opt/EmbedSky/bootloader/arch/s3c2440/mmu.c
opt/EmbedSky/bootloader/arch/s3c2440/nand_read.c
opt/EmbedSky/bootloader/arch/s3c2440/proc.c
opt/EmbedSky/bootloader/arch/s3c2440/smdk.c
opt/EmbedSky/bootloader/arch/s3c2440/smdk2440_test.c
opt/EmbedSky/bootloader/arch/s3c2440/test.c
opt/EmbedSky/bootloader/init/
opt/EmbedSky/bootloader/init/version.c
opt/EmbedSky/bootloader/init/main.c
opt/EmbedSky/bootloader/.ChangeLog.swp
opt/EmbedSky/bootloader/COPYING
opt/EmbedSky/bootloader/ChangeLog
opt/EmbedSky/bootloader/Makefile
opt/EmbedSky/bootloader/Rules.make
```

The following steps illustrate how to compile bootloader.

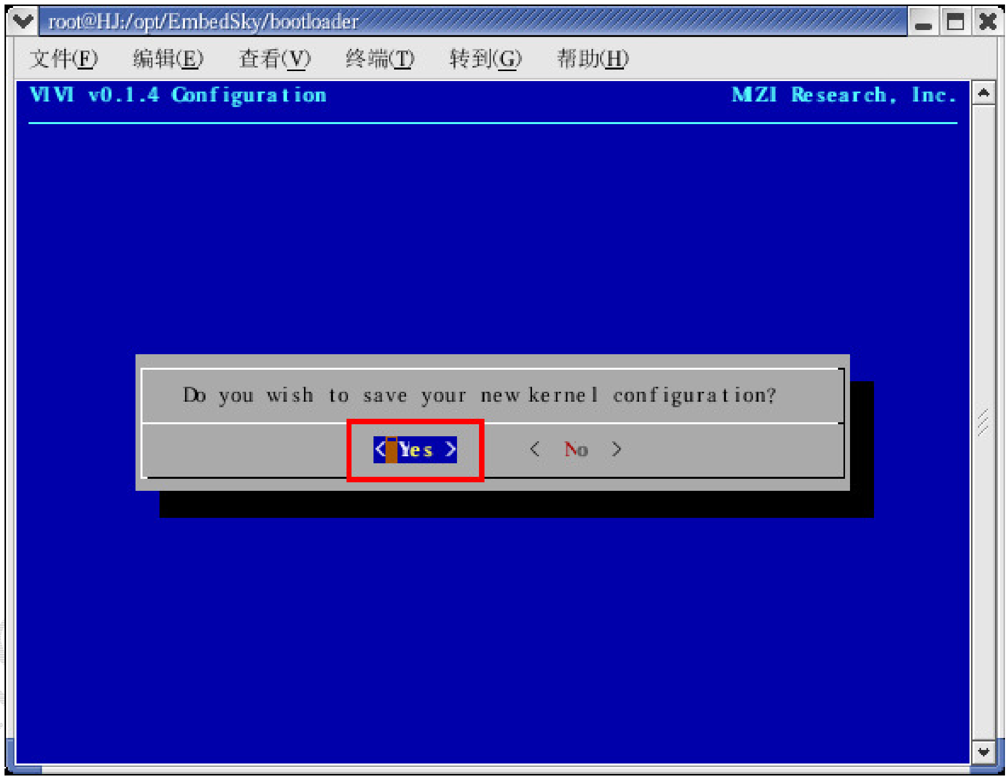
Step1, configure bootloader. Get into the directory of bootloader and input the command “make menuconfig”:



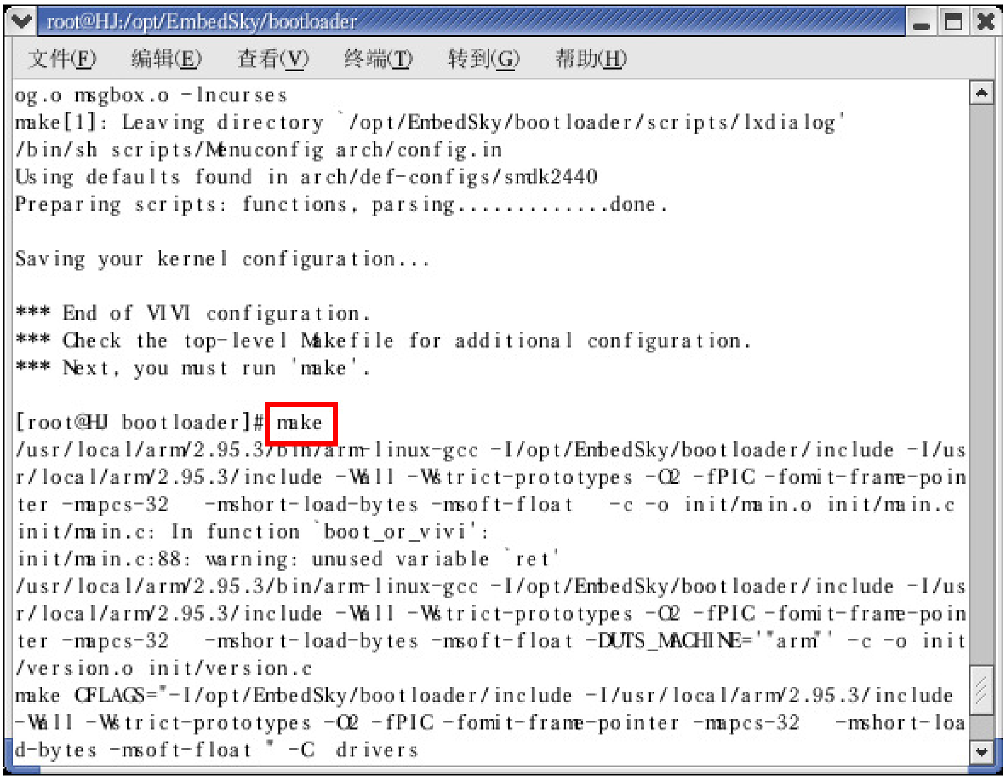
Step2, get into bootloader configuration interface and use the default parameter. Select “<Exit>” to exit:



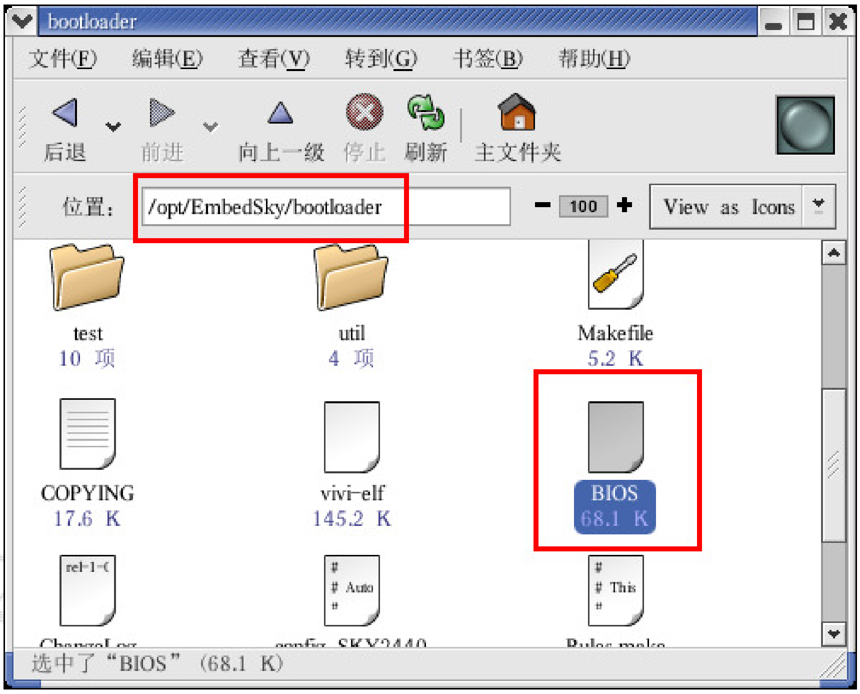
Select “<Yes>” to save the configuration and continue:



Step3, compile bootloader. Enter the command “make” and press return-key to start compiling:



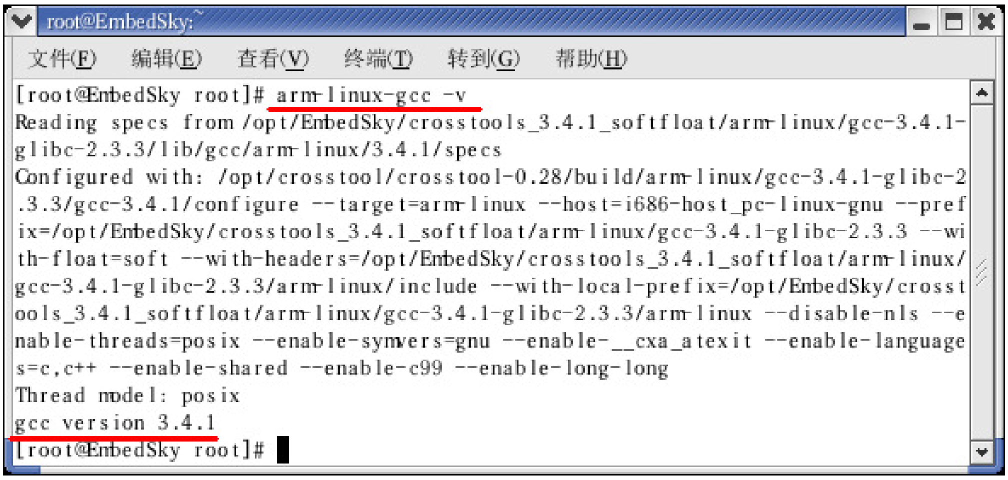
Step4, the file “BIOS” will appear under the directory “/opt/EmbedSky/bootloader/” after compiling:



Burn the file “BIOS” to platform and then it can be used to boot Linux.

4. 2 Compiling Linux-2. 6 kernel

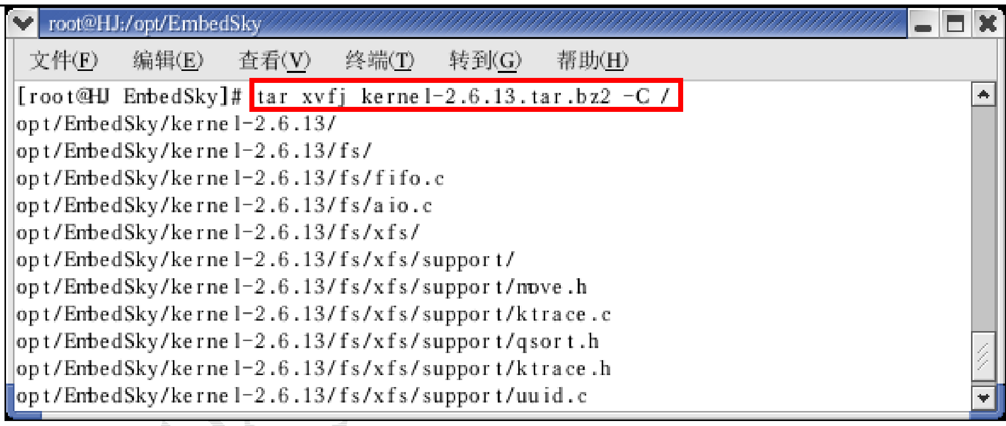
Caution: The cross-compiler of version 3.4.1_softfloat is needed when compiling Linux kernel of version 2.6. Make sure the compiler of correct version has been installed successfully.



If the compiler version 3.4.1_softfloat has been successfully installed, the upper information above the red line will appear; Otherwise consult “2.5.1 节” to install the compiler.

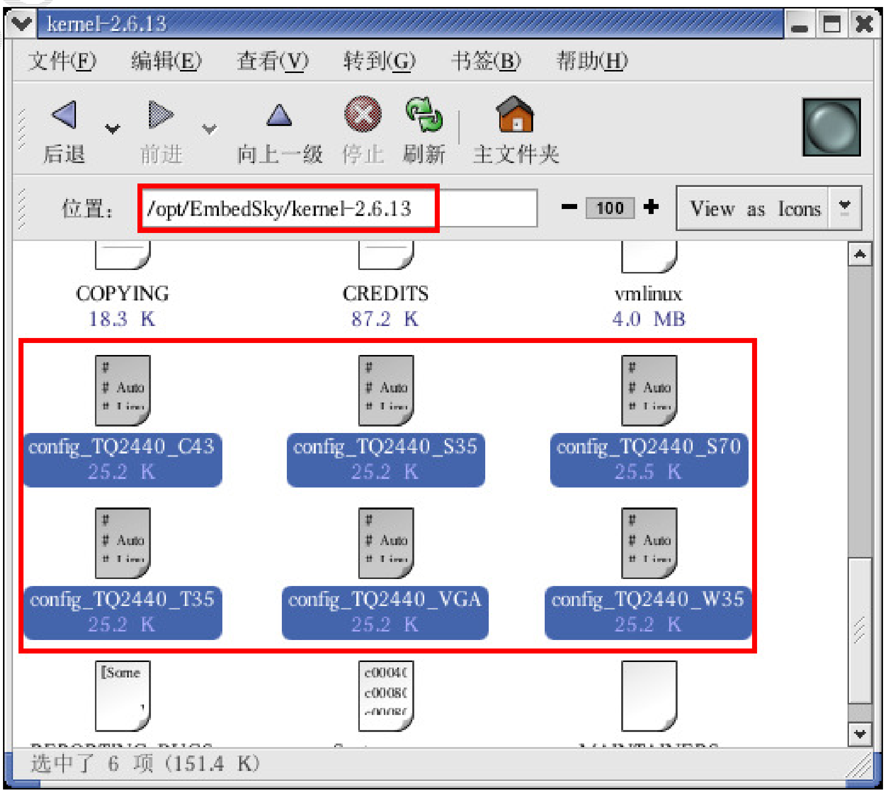
4. 2. 1 Use configuration file of EmbedSky to compile Linux kernel

The Linux kernel source code is under the directory “Linux/ kernel-2.6.13.tar.bz2”. Execute the command “#tar xvfj kernel-2.6.13.tar.bz2 –C /” to decompress it to the directory “/opt/EmbedSky/kernel-2.6.13”:



The default files are under the decompression directory:

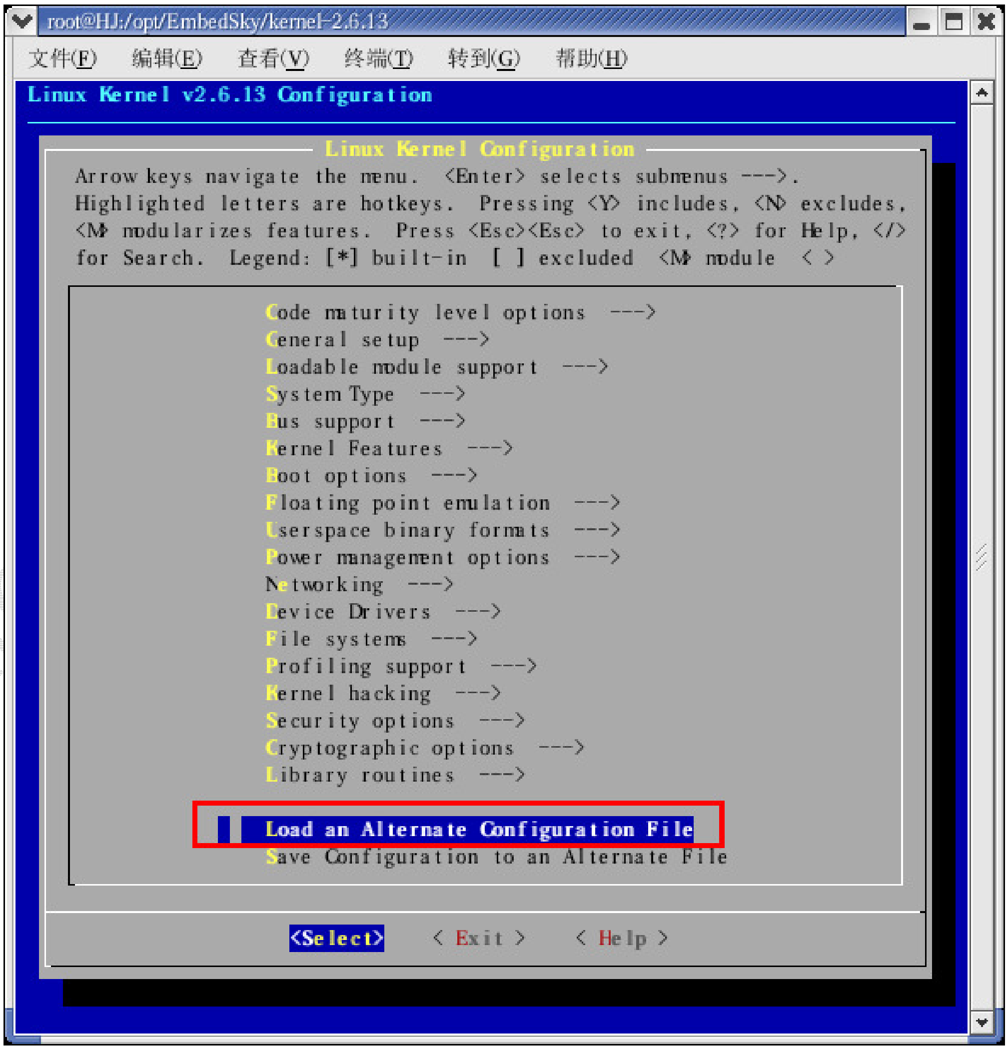
- config_TQ2440_S35 is the Samsung 3.5 inch LCD default configuration file.
- config_TQ2440_S70 is the Samsung 7 inch LCD default configuration file.
- config_TQ2440_T35 is the Toshiba 3.5 inch LCD default configuration file.
- config_TQ2440_W35 is the Donghua 3.5 inch LCD default configuration file.



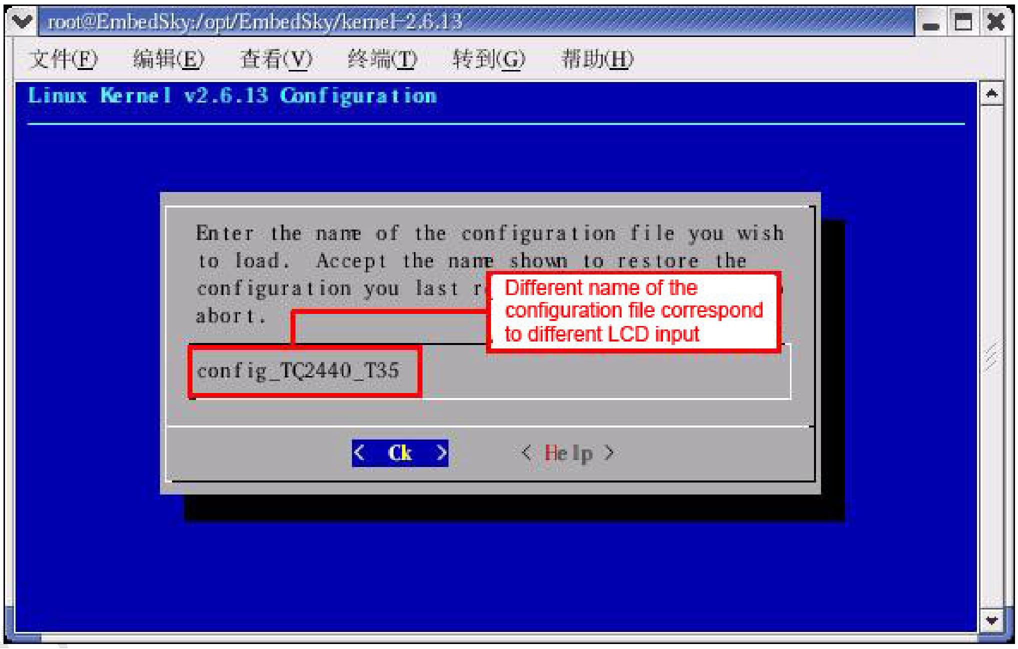
The following steps illustrate the process compiling Linux kernel:

Step1, input “make menuconfig” to start configuring Linux kernel:

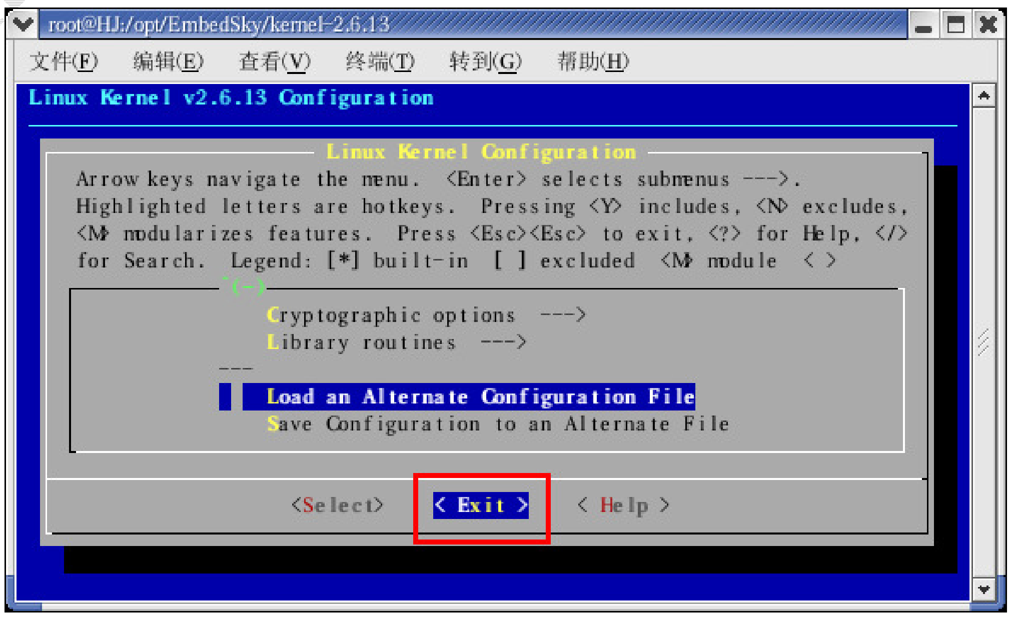
Step2, select “Load an Alternate Configuration File”:

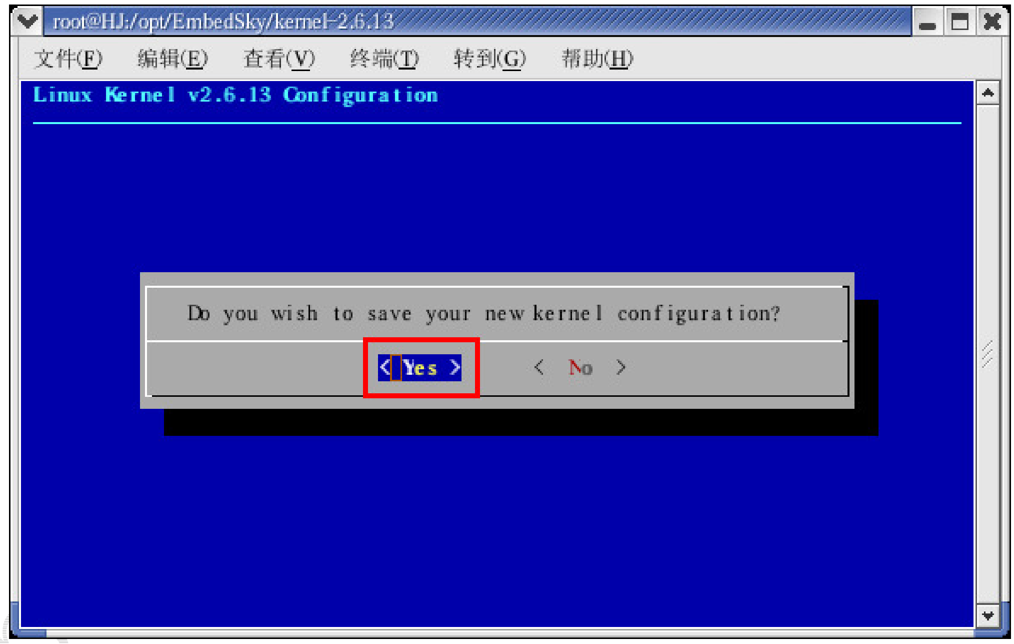


Step3, input the configuration file name according to the LCD type you are using. Here we use the configuration file of Toshiba 3.5 inch LCD for example. Click “OK” after input complete.

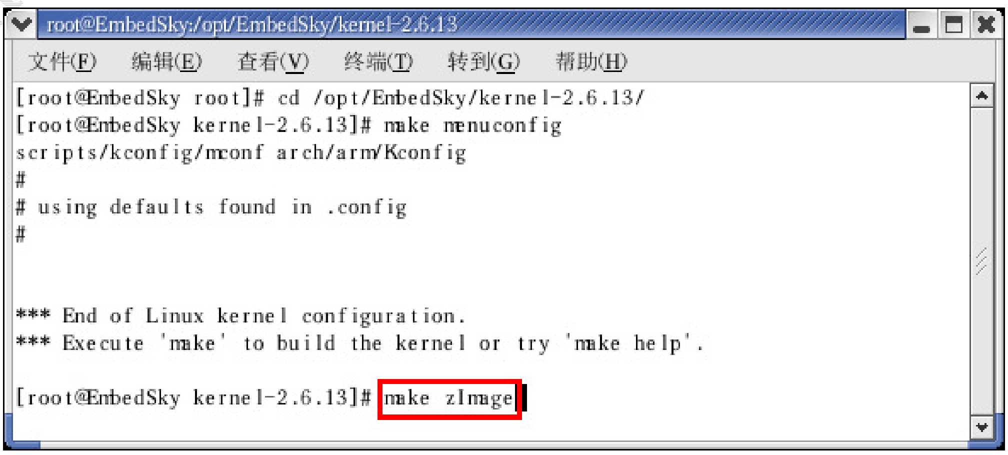


Step4, go back to main menu. Select “<Exit>” to save the configuration and exit:

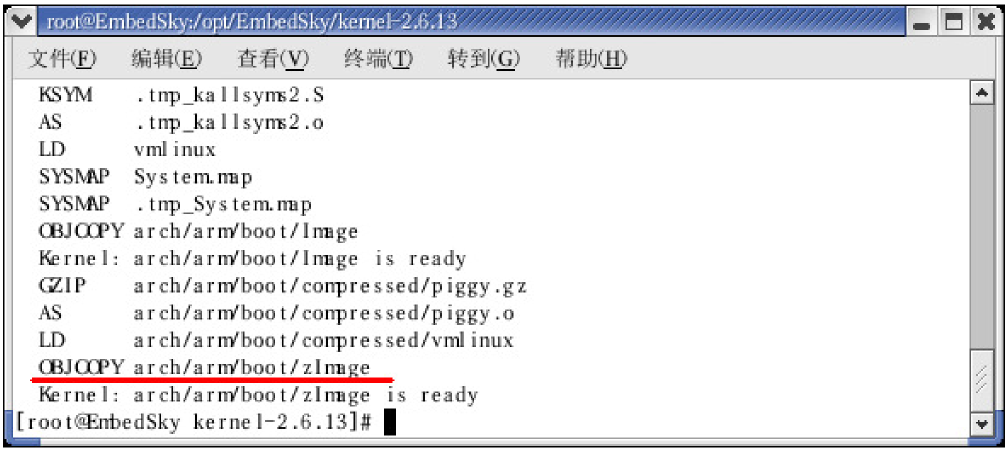




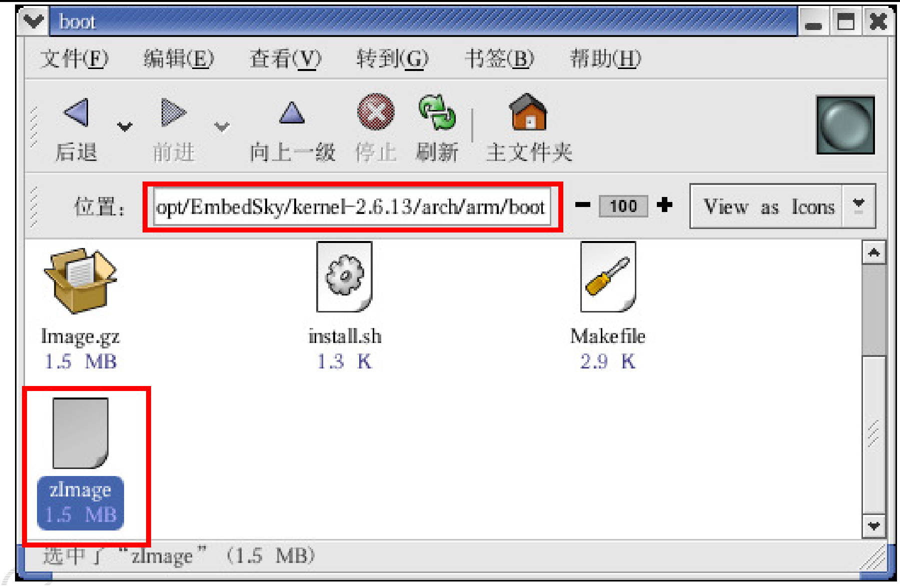
Step6, input “make zImage” and press return-key to start compiling:



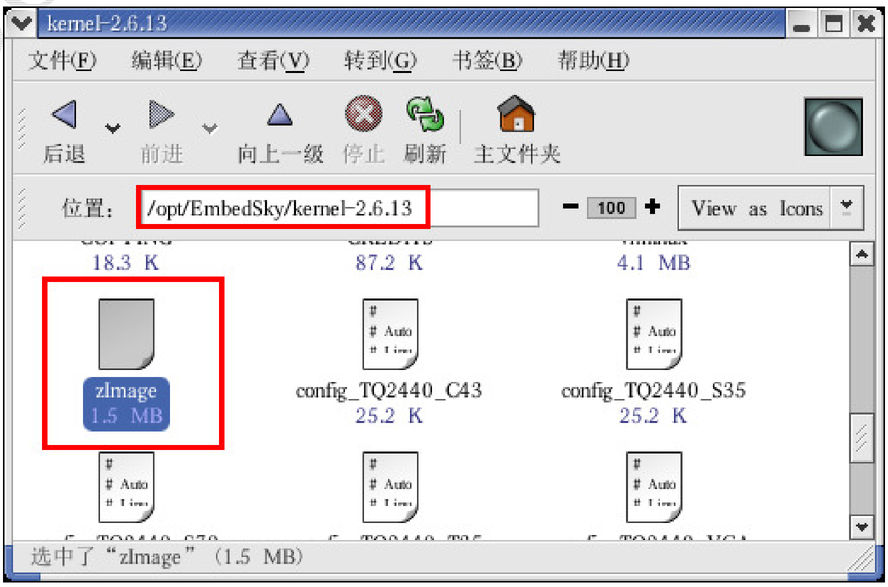
Step7, the compiling is complete:



The kernel image file “zImage” is automatically created under the directory “/opt/EmbedSky/kernel-2.6.13/arch/arm/boot/” or “/opt/EmbedSky/kernel-2.6.13/” after compilation:



And

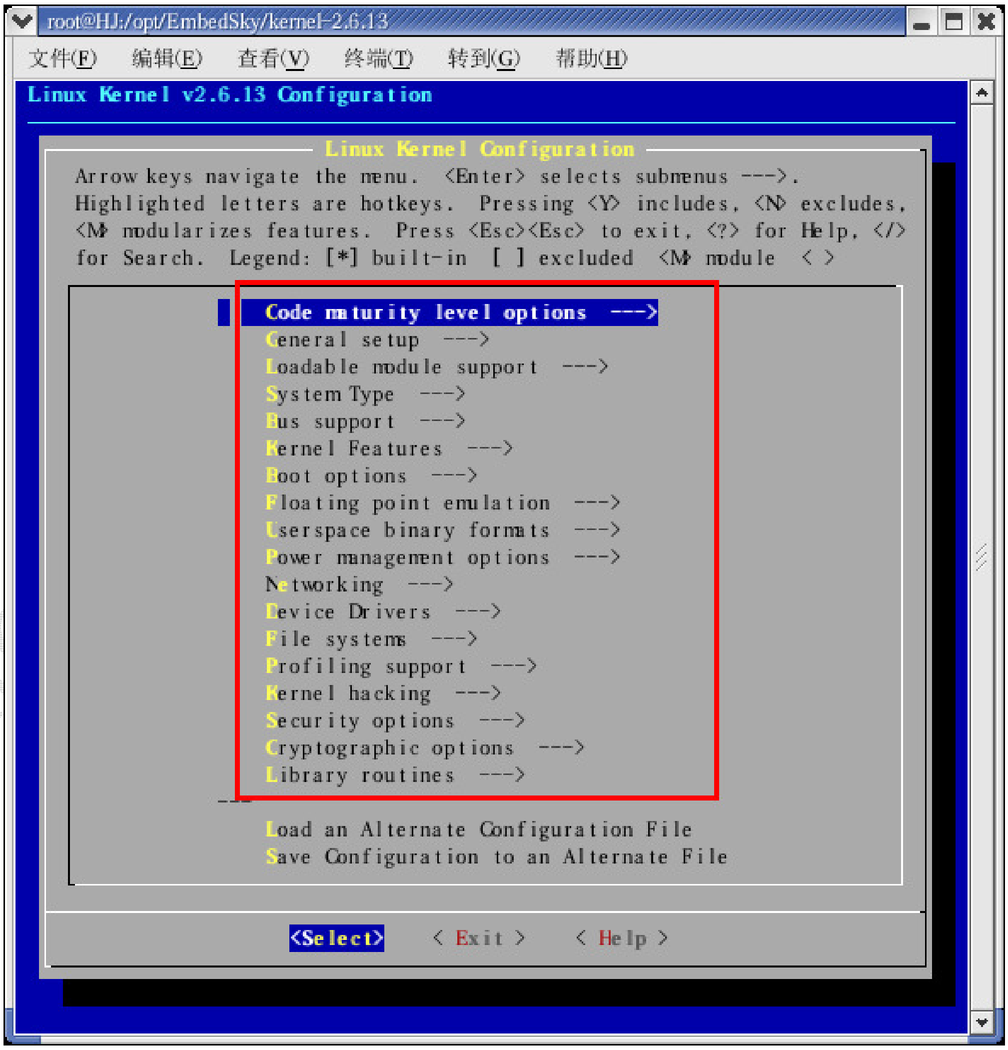


Burn the kernel image file to platform.

4. 2. 2 Customizing Linux kernel

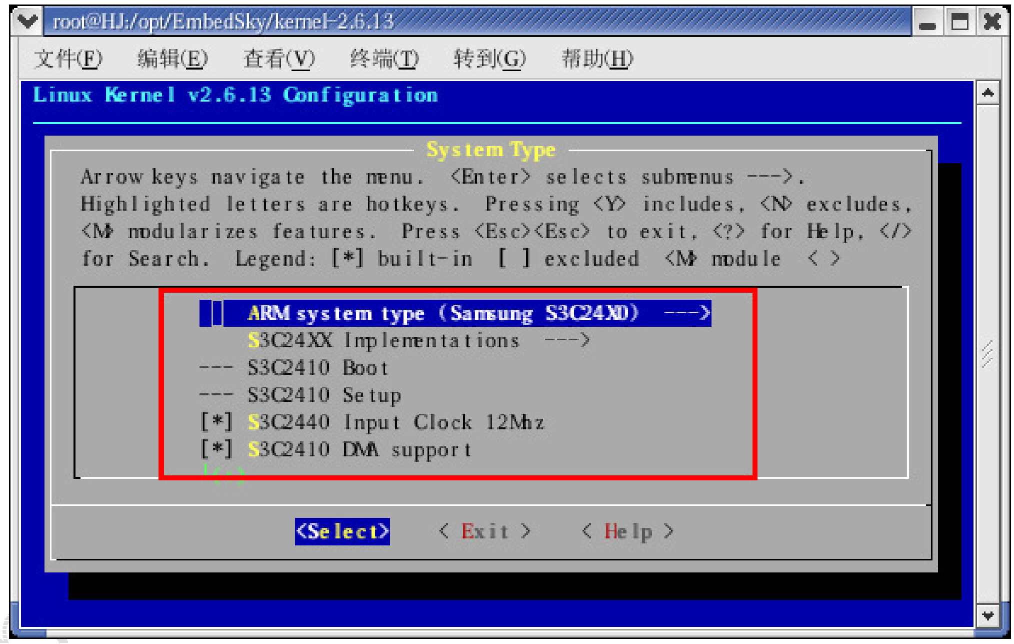
In previous introductions, we use the default files to configure the Linux kernel. Here we introduce more other knowledge about configuring Linux kernel to enhance your understanding.

Execute “make menuconfig” to enter into Linux kernel configuration main menu:



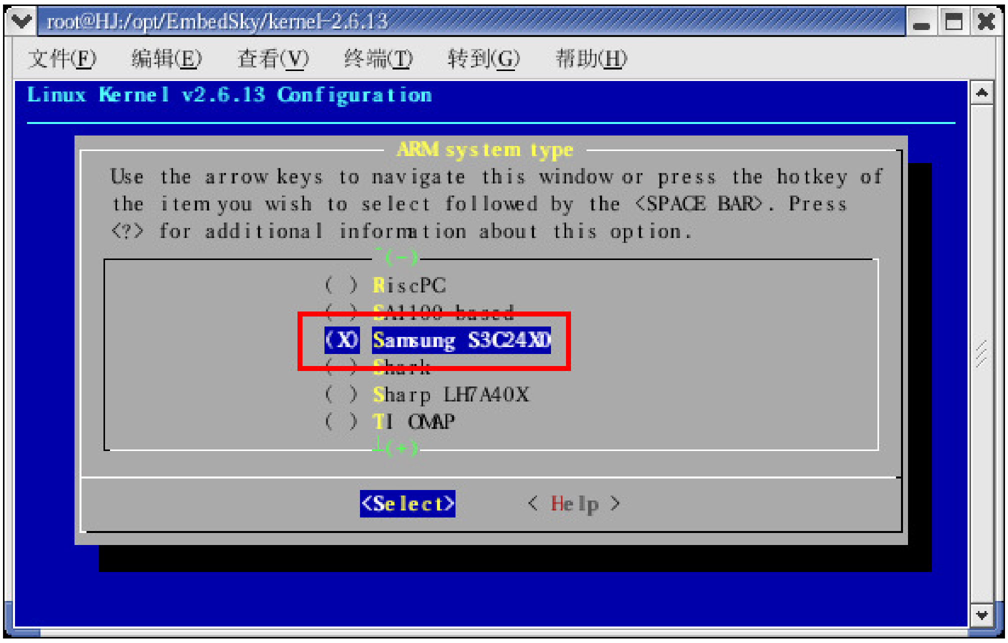
- **Configure CPU:**

Select "System Type" in main menu and press return-key:

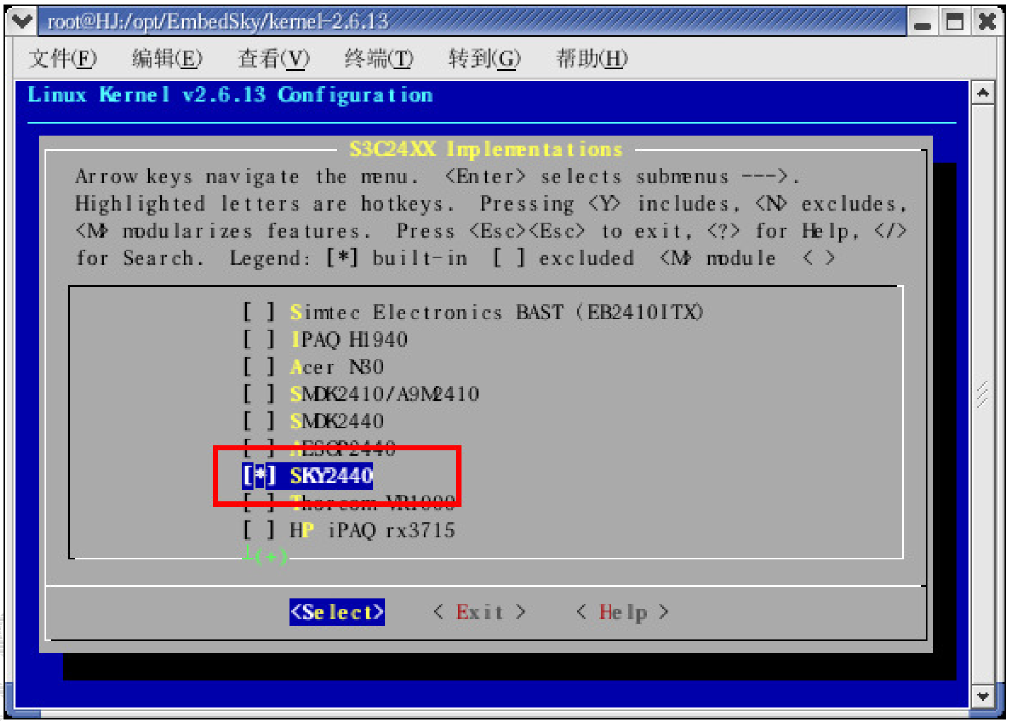


The upper diagram displays the options of S3C2410, S3C2440, S3C24X0 and S3C24XX. The ARM chips of S3C24X0 serial always have the same register address and setting patterns. Therefore we do not do any configuration to these 2 kinds of CPU in version 2.6 Linux kernel any more.

In the menu "ARM system type (Samsung S3C24X0)", select the CPU type "Sansung S3C24X0" and press return-key to continue:

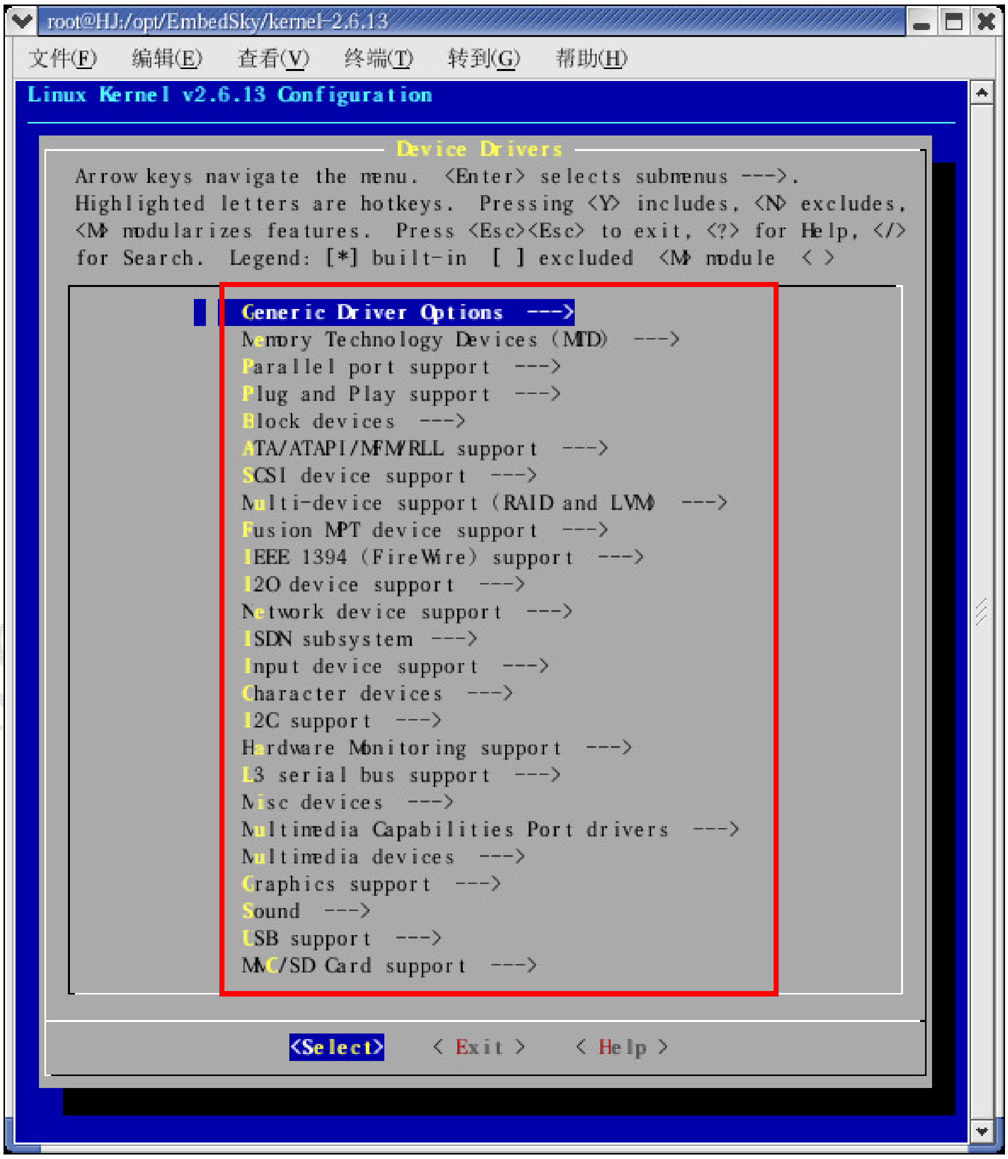


Select the platform type "SKY2440" in menu "S3C24XX Implementations". After that, select "Exit" and press return-key to continue:



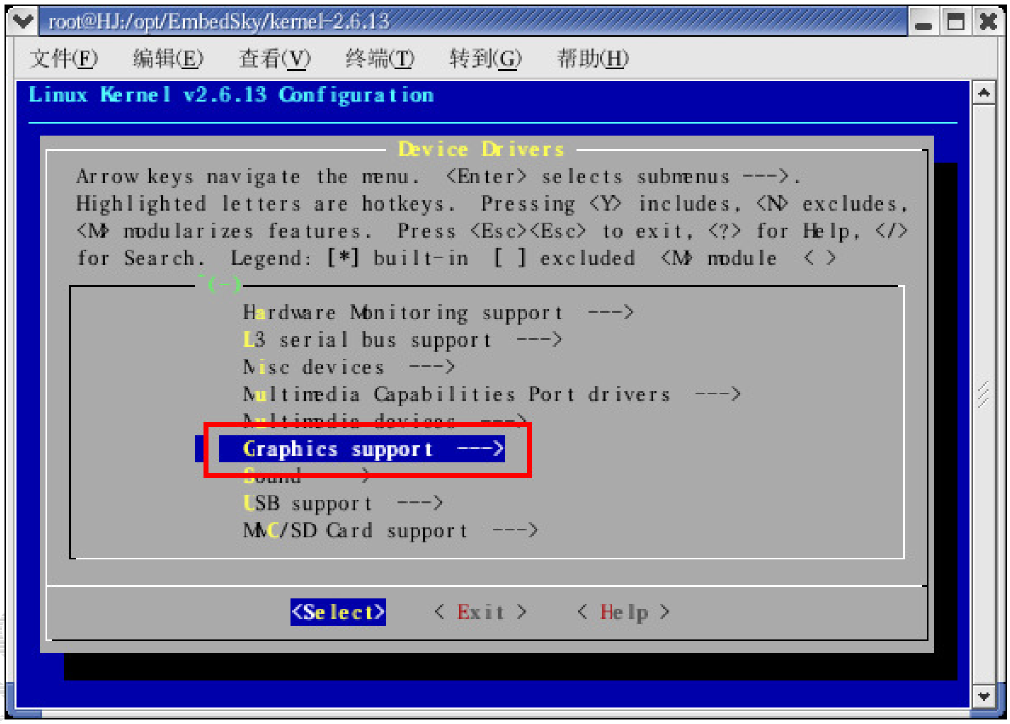
The type of our Development Board is SKY2440, which is corresponding to the file “sky2440.c” under the directory “arch/arm/mach-s3c2410/”.

All the device configuration sub-menus are under the main menu “Device Drivers”.

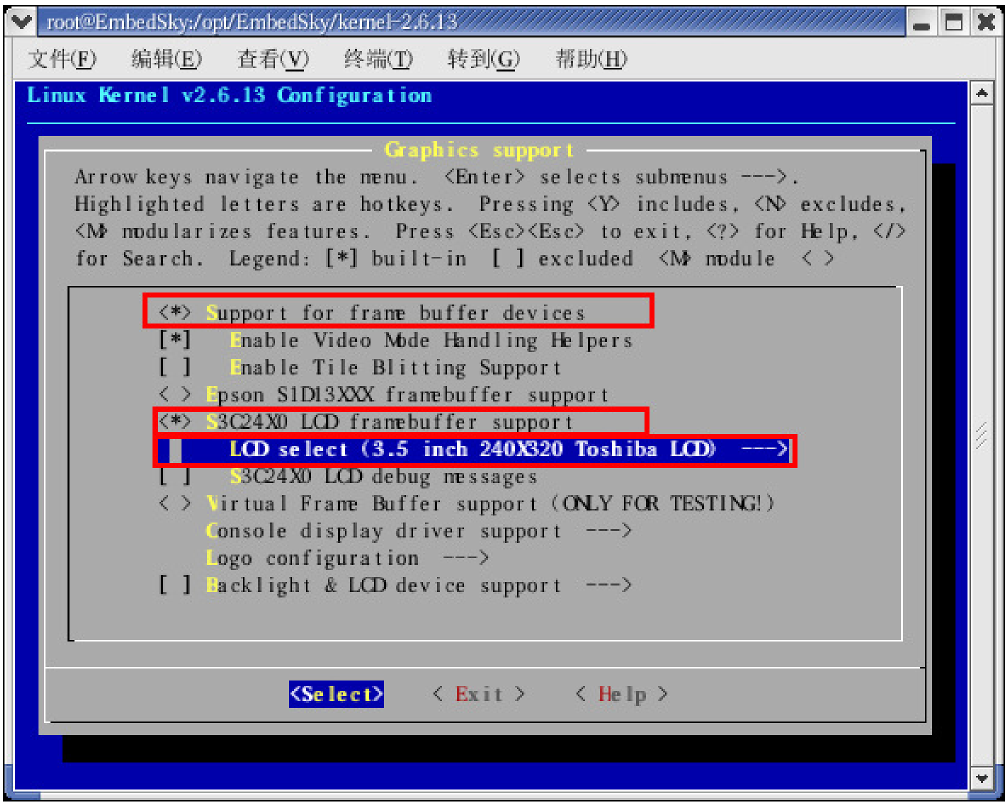


➢ **Configure LCD:**

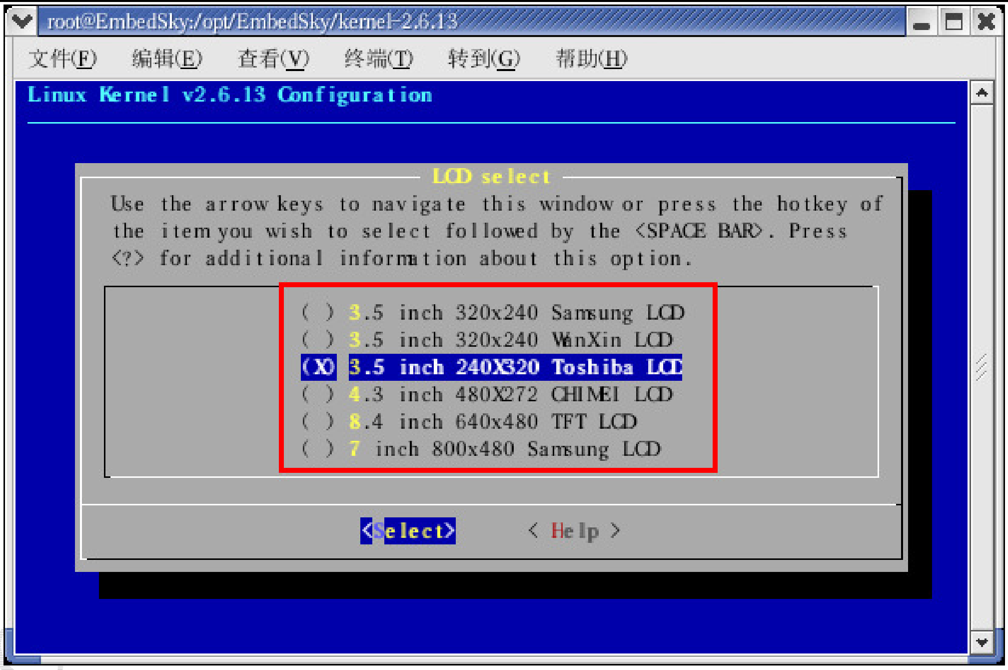
Select “Graphics support” in the menu “Device Drivers”, and press return-key to continue:



Select “Support for frame buffer devices”, “S3C24X0 LCD framebuffer support” and “LCD select” in sequence:



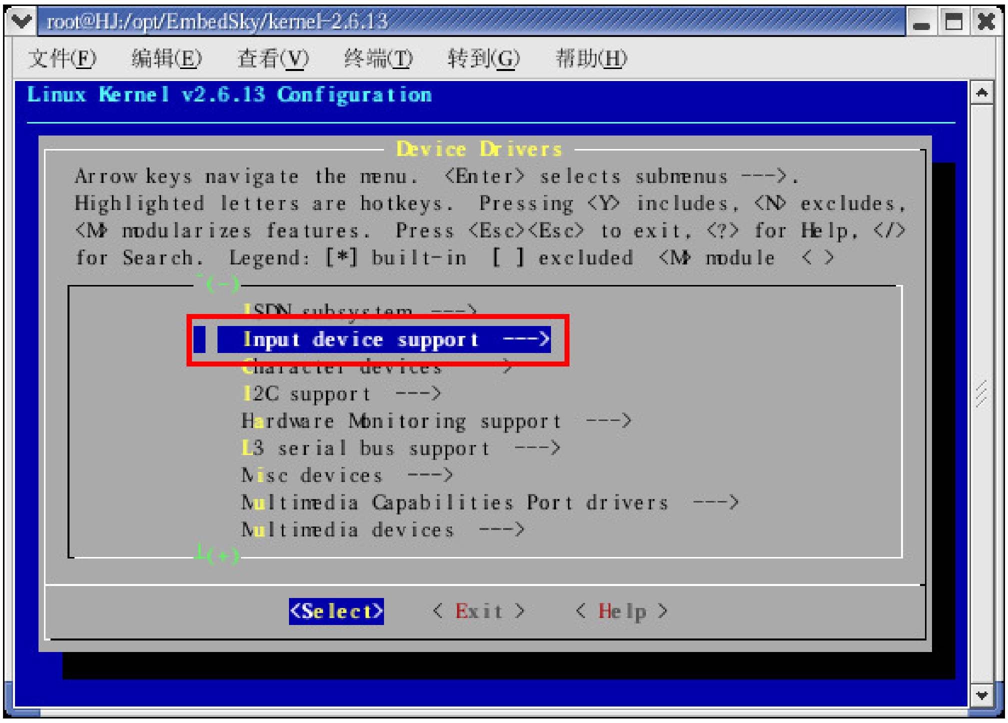
Select the LCD driver in “LCD select” menu. Here we select “3.5 inch 240*320 Toshiba LCD”:



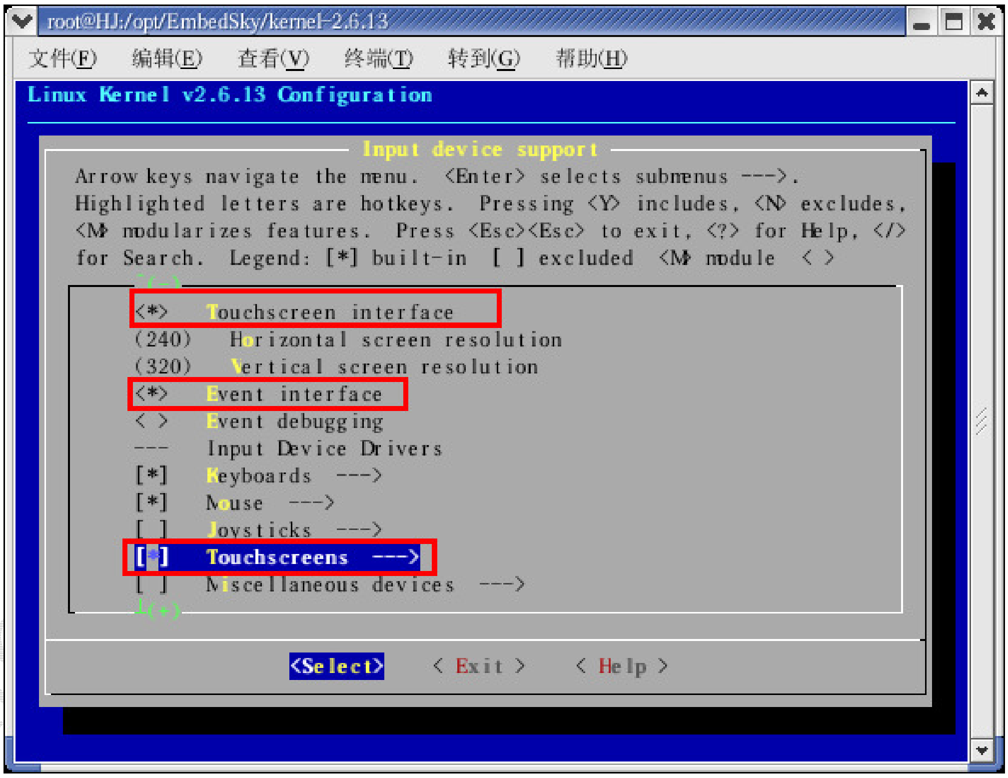
After selection, press return-key to go back to the upper menu. And select "<Exit>" to go back to the menu "Device Drivers".

➢ **Configure touch-screen:**

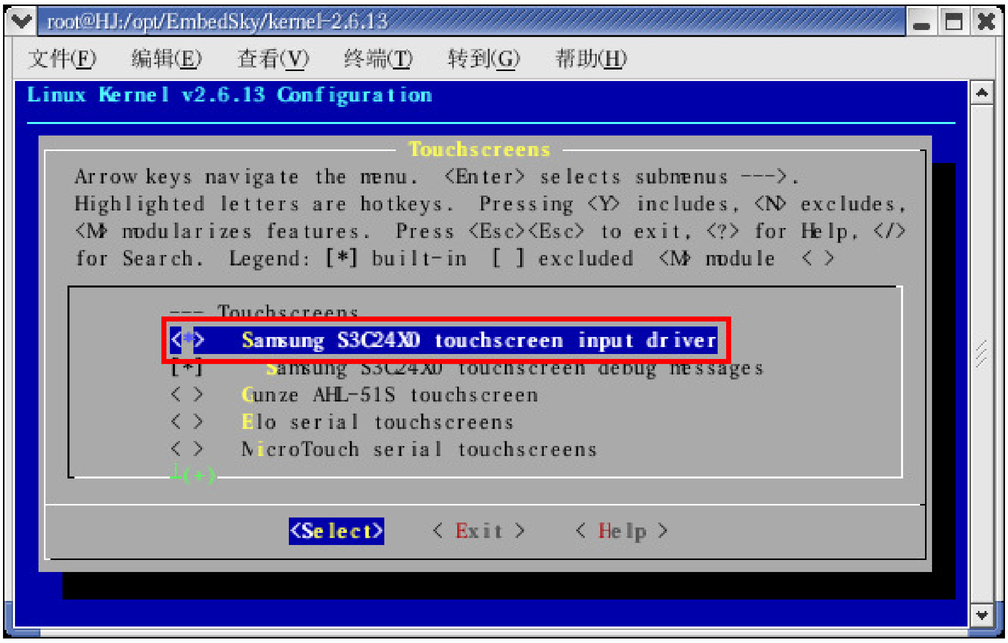
Select "Input device support" in the menu "Device Drivers" and press return-key to continue:



Select "Touchscreen interface", "Event interface" and "Touchscreens" in sequence to continue:



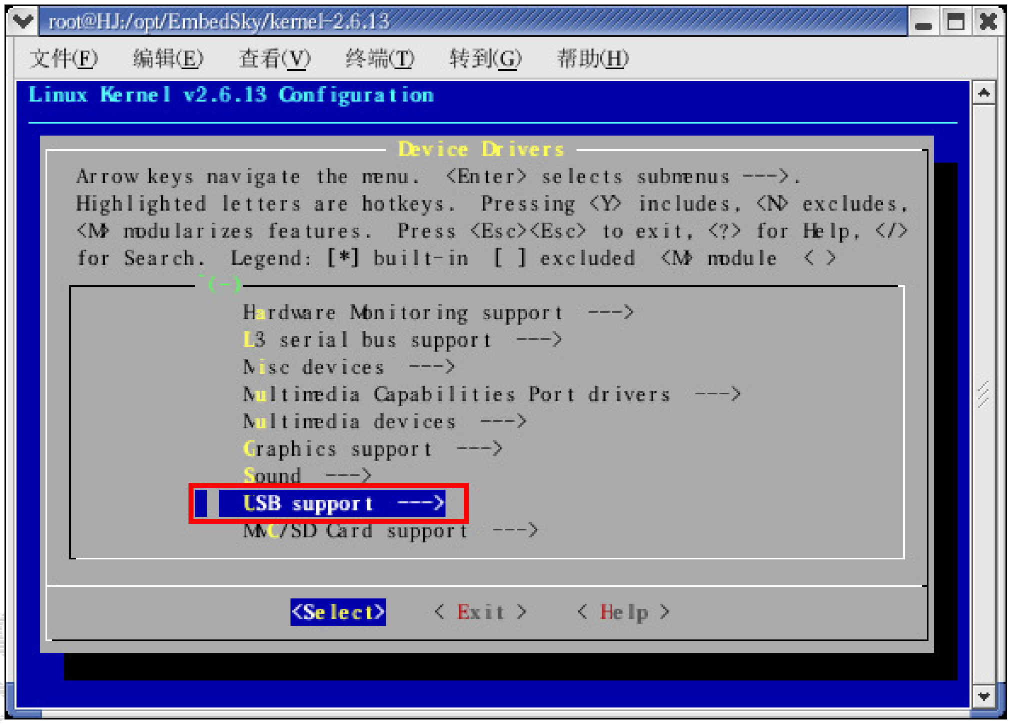
Select “Samsung S3C24X0 touchscreen input driver” in the sub-menu “Touchscreens” and continue:



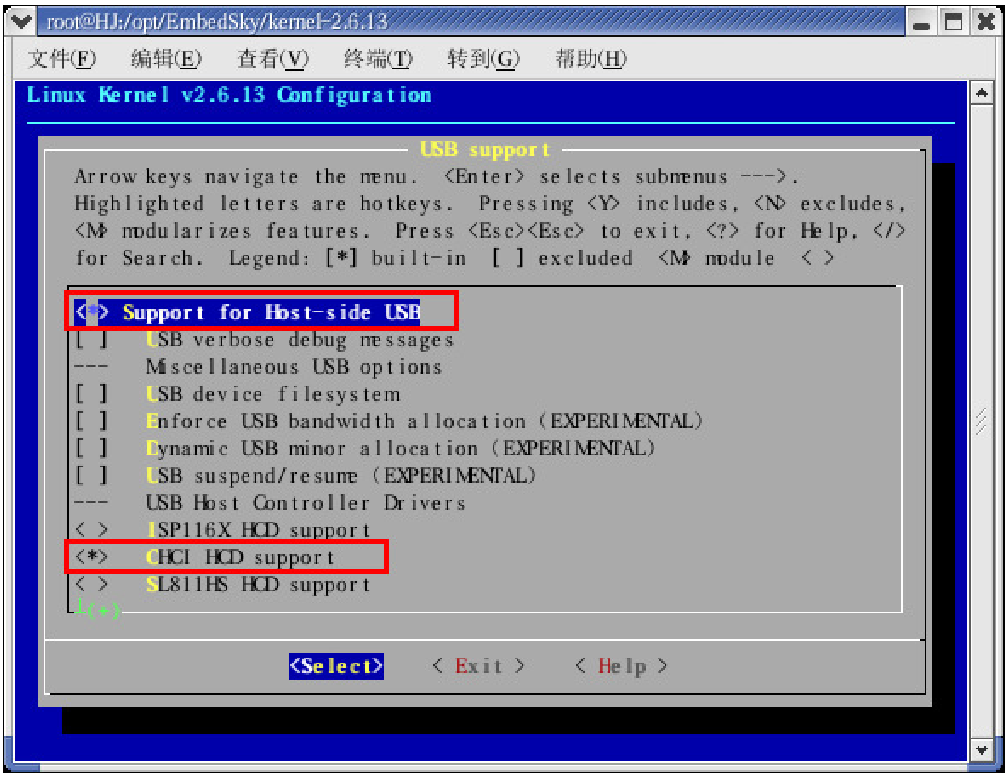
Select “<Exit>” to go back to the upper menu, and select “<Exit>” to go back to the menu “Device Drivers”.

➢ **Configure USB mouse & keyboard:**

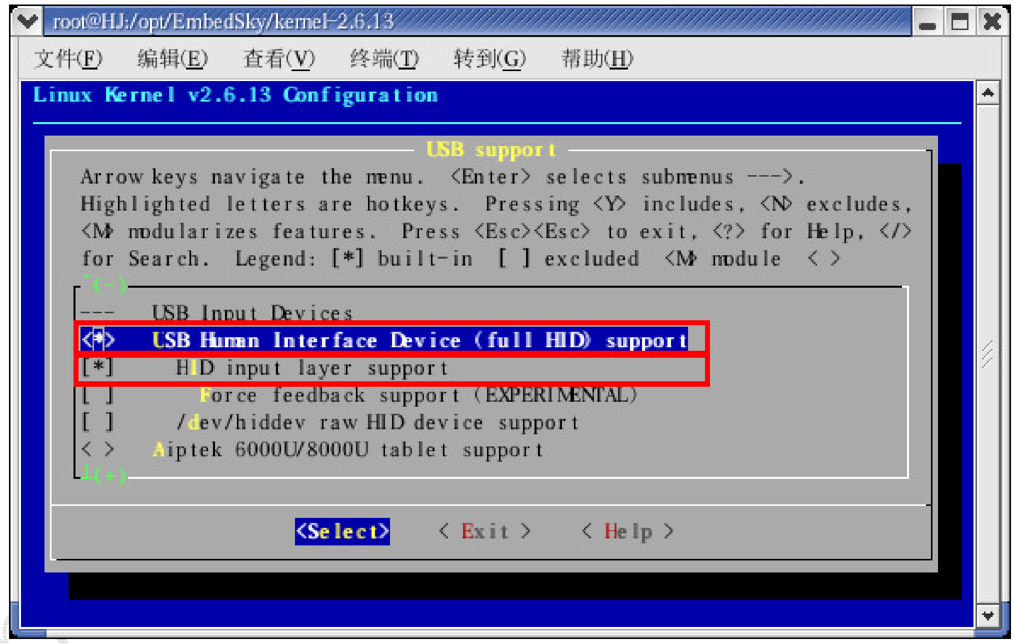
Select “USB support” in the menu “Device Drivers” and press return-key to continue:



Find and select “Support for Host-side USB” and “OHCI HCD support”:



Select the USB mouse and keyboard option “USB Human Interface Device (full HID) support” and “HID input layer support”:

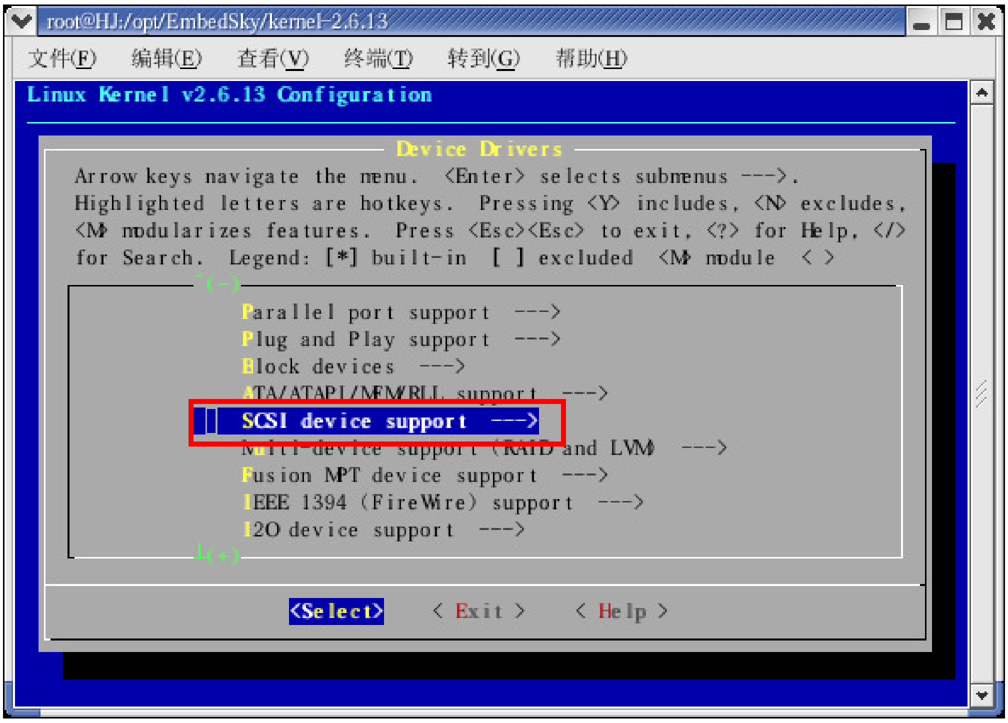


Select “<Exit>” to go back to the menu “Device Drivers”.

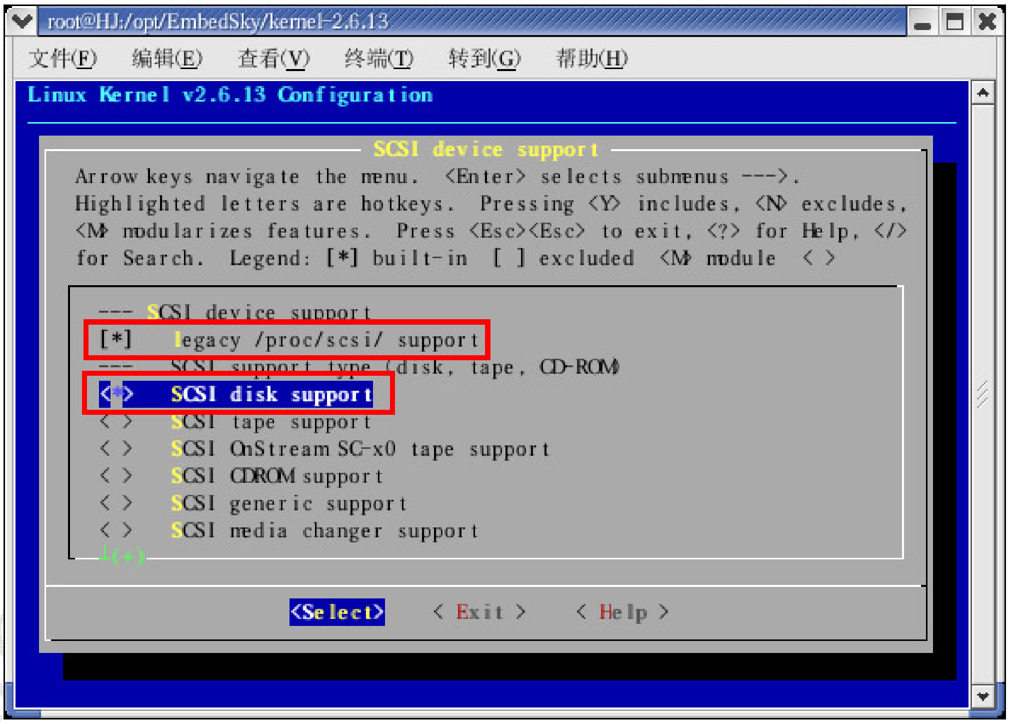
- **Configure USB memory:**

The instruction set SCSI is needed.

Select “SCSI device support” in the menu “Device Drivers” and press return-key to continue:

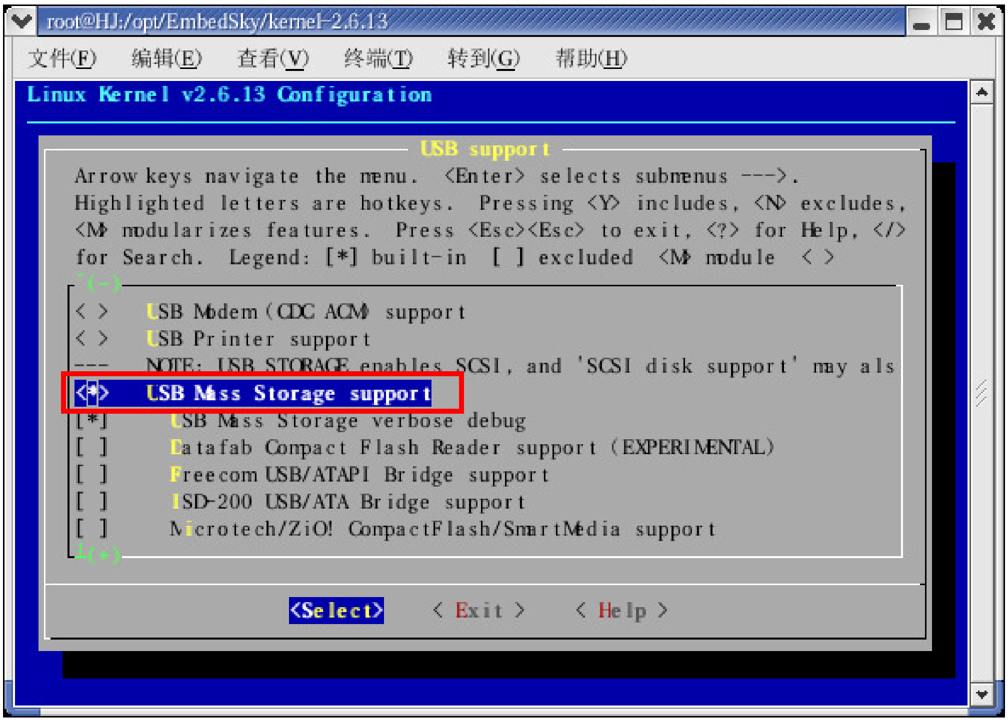


Select “legacy /proc/scsi support” and “SCSI disk support”:



Select “<Exit>” to go back to the menu “Device Drivers”.

Select “USB support” in the menu “Device Drivers” and press return-key to enter into “USB support” menu. And then select “USB Mass Storage support”:

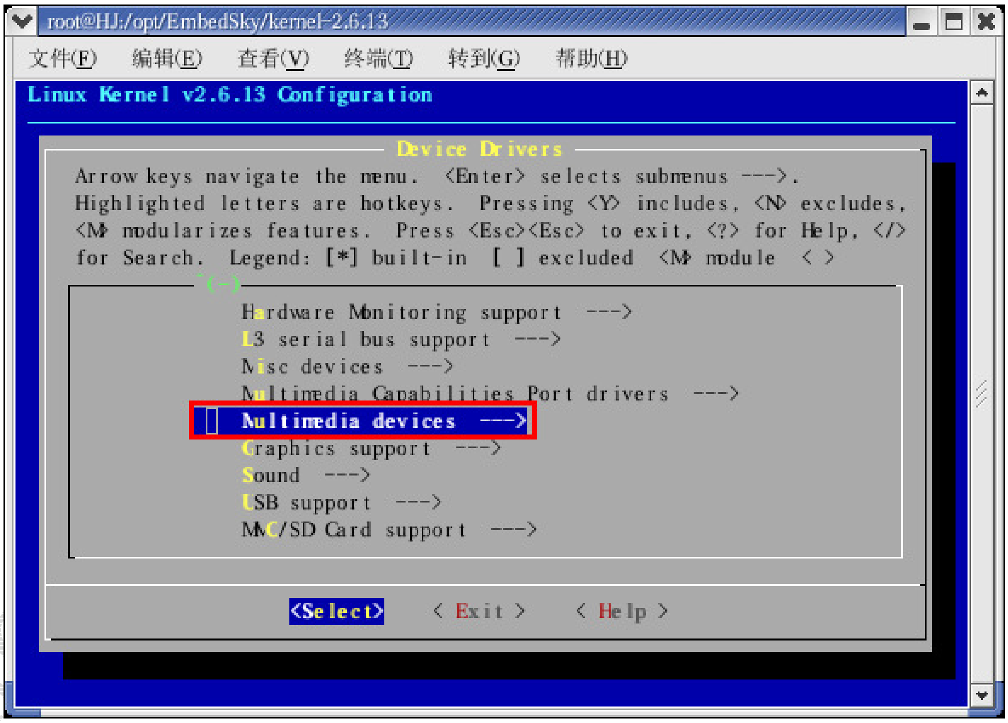


Select “<Exit>” to go back to the menu “Device Drivers”.

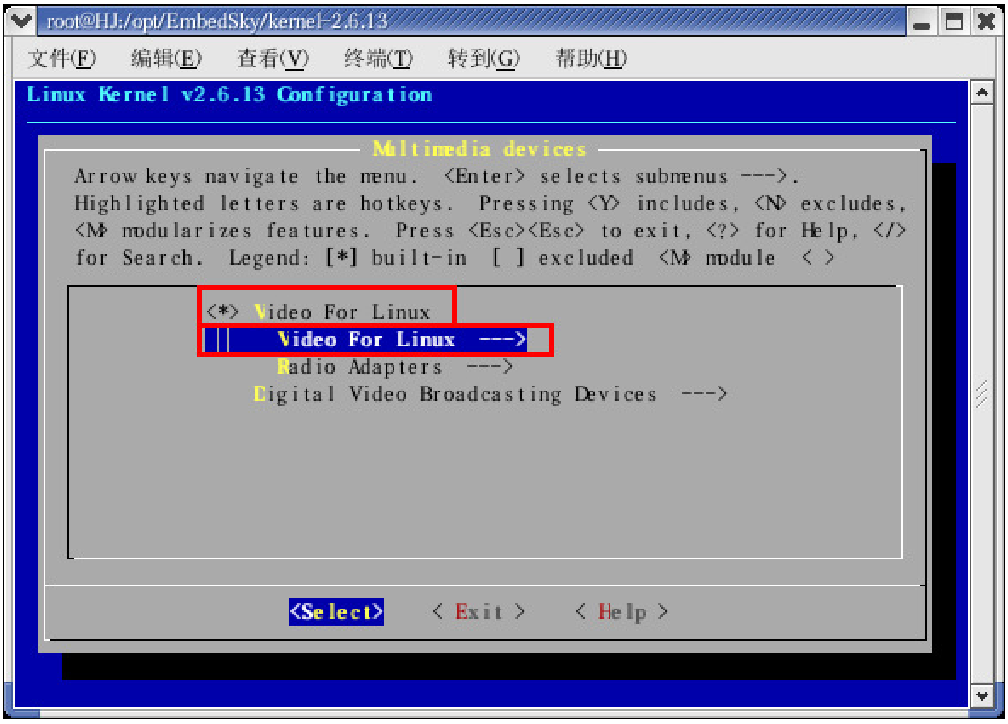
➢ **Configure USB camera:**

The USB camera needs the support of V4L.

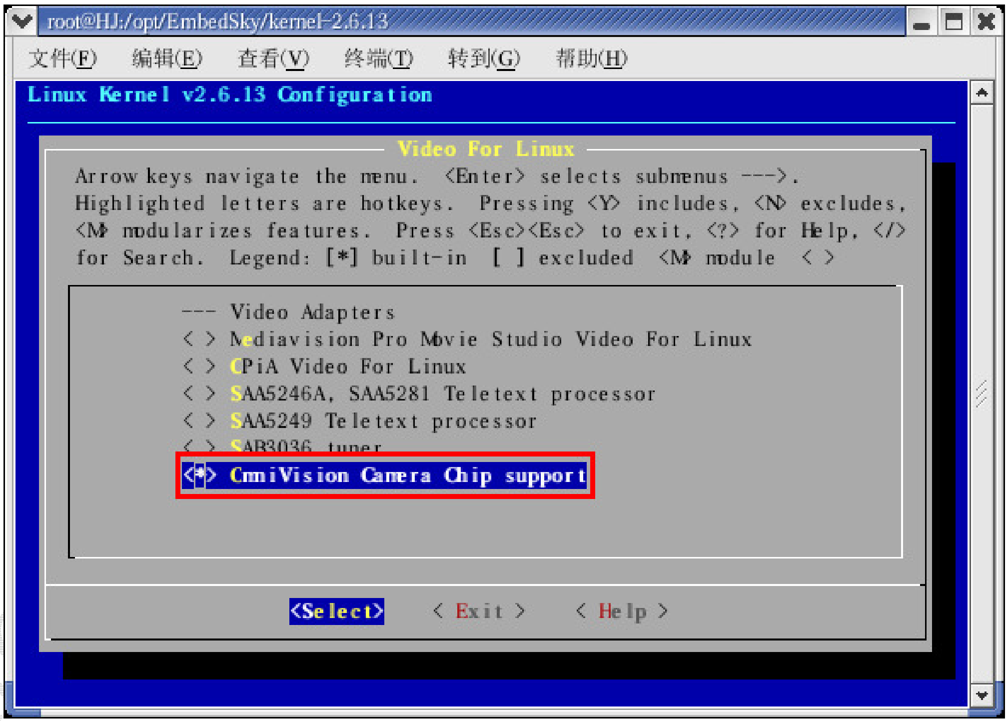
Select “Multimedia devices” in the menu “Device Drivers” and press return-key to continue:



Select “Video For Linux” and press return-key to continue:



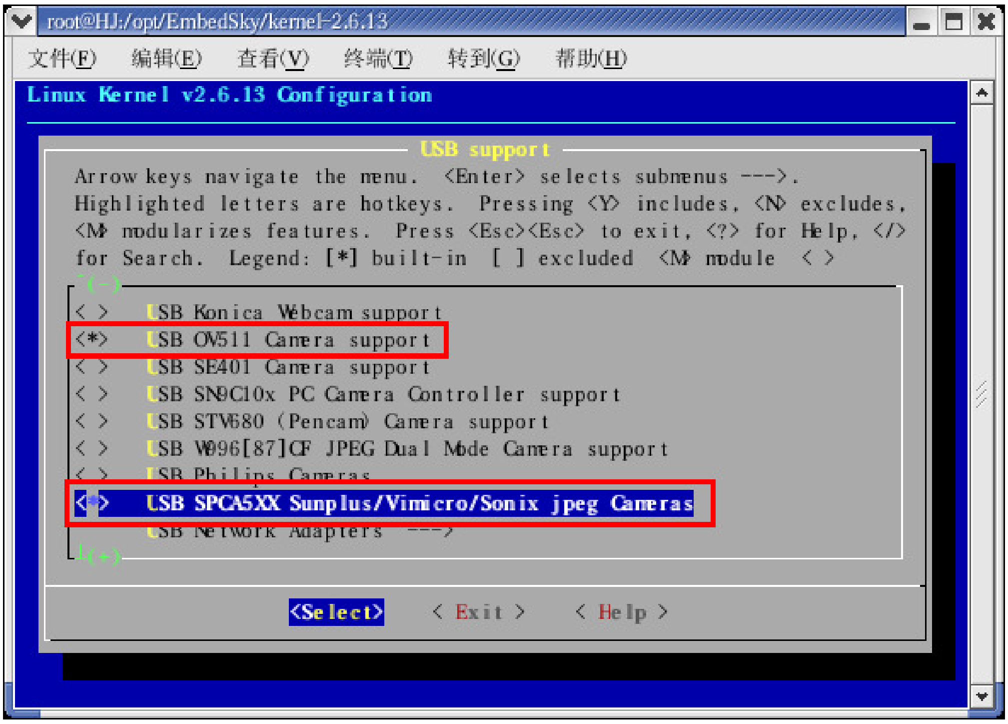
Select “OmniVision Camera Chip support” in the menu “Video For Linux”:



Select “<Exit>” to go back to the menu “Multimedia devices”. And then select “<Exit>” again to go back to the menu “Device Drivers”

Select “USB support” in the menu “Device Drivers” and get into the sub-menu “USB support”. Find and select “USB OV511 Camera support” and “USB SPCA5XX Sunplux/Vimicro/Sonix jpeg Cameras”:

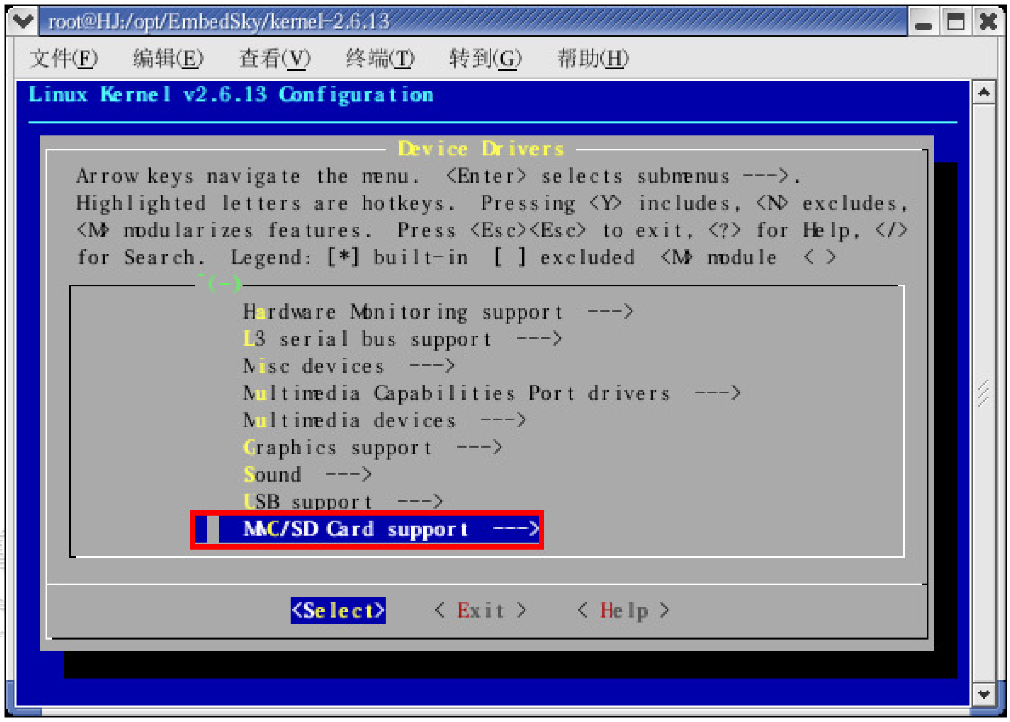
(OV511 supports the cameras based on OV511 chip, and SPCA5XX supports the cameras based on Smics micro 301 serials chips which own more than 70 percent market share.)



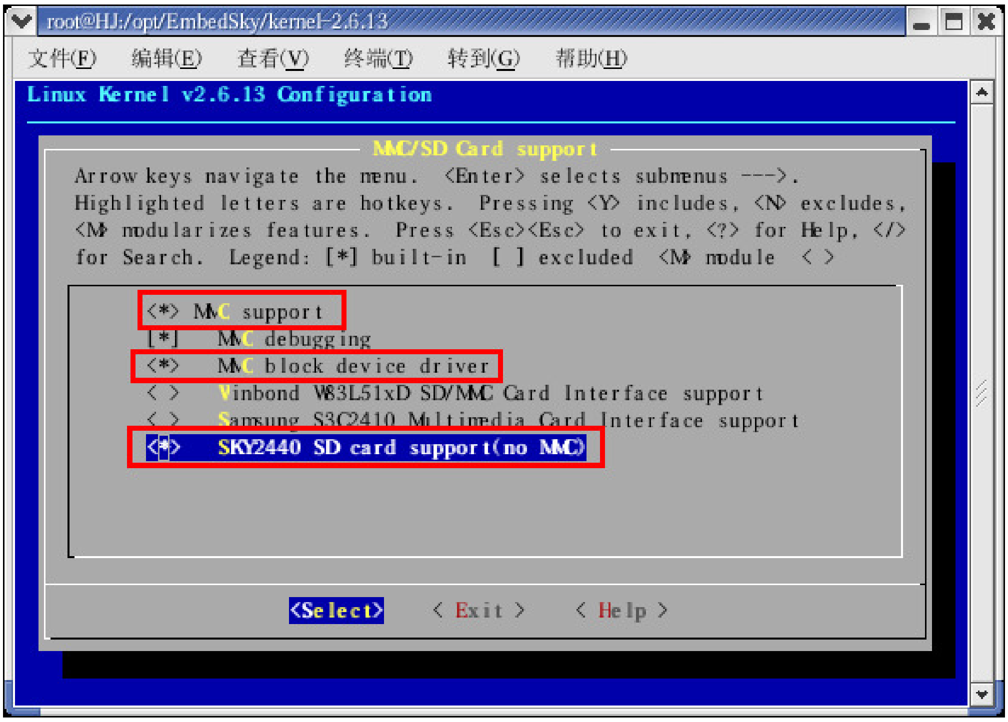
Select “<Exit>” to go back to the menu “Device Drivers”.

➢ **Configure SD card:**

Select “MMC/SD Card support” in the menu “Device Drivers”, and press return-key to get in:



Select “MMC support”, “MMC block device driver” and “TQ2440 SD card support (no MMC)” in the sub-menu “MMC/SD Card support”:

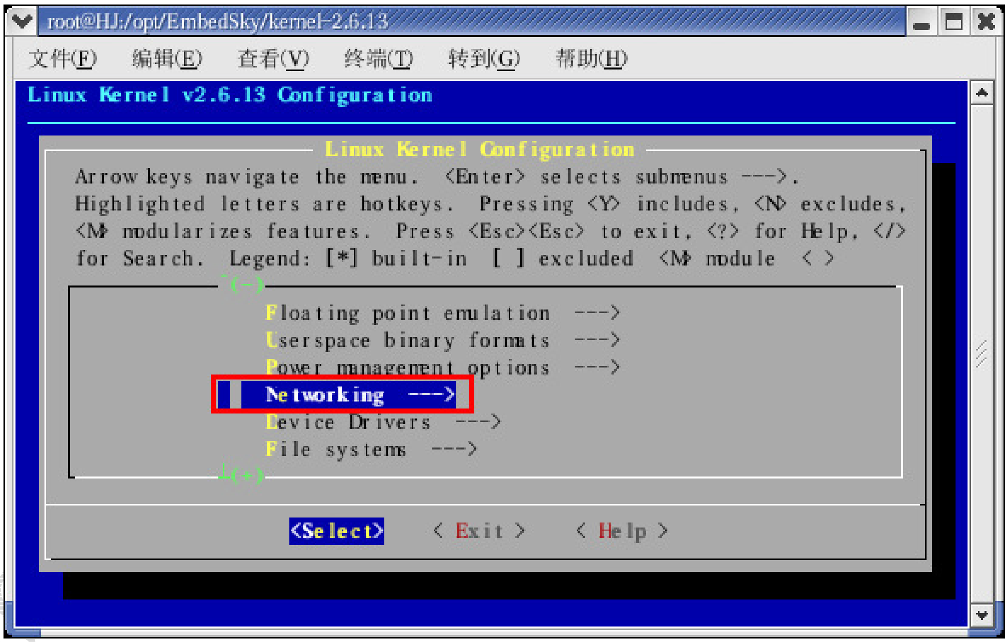


Caution: the upper configuration only contains SD card driver, and no MMC card support is included.

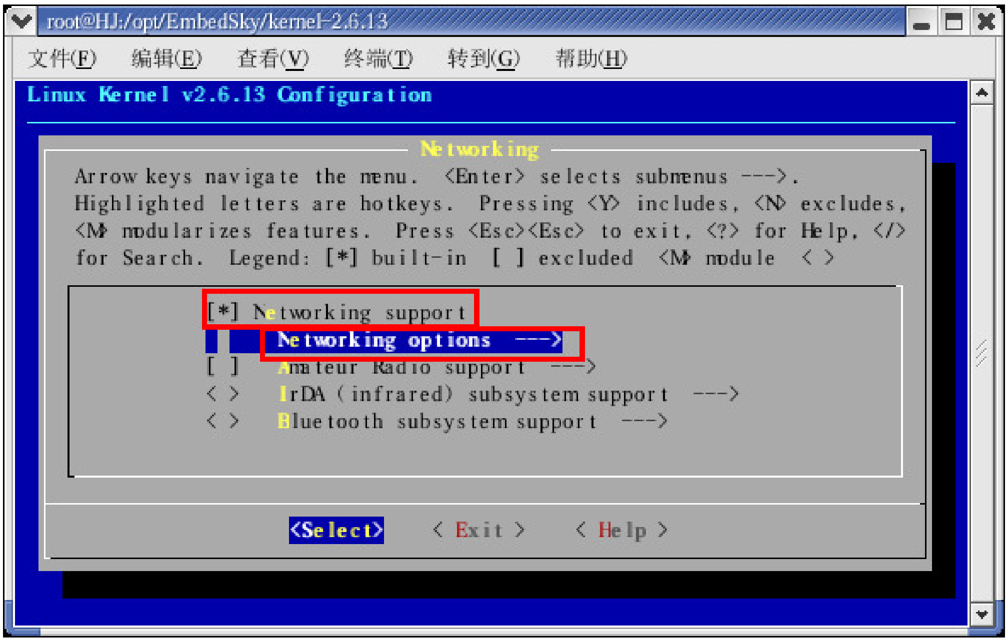
Select “<Exit>” to go back to the menu “Device Drivers”.

➢ **Configure network card:**

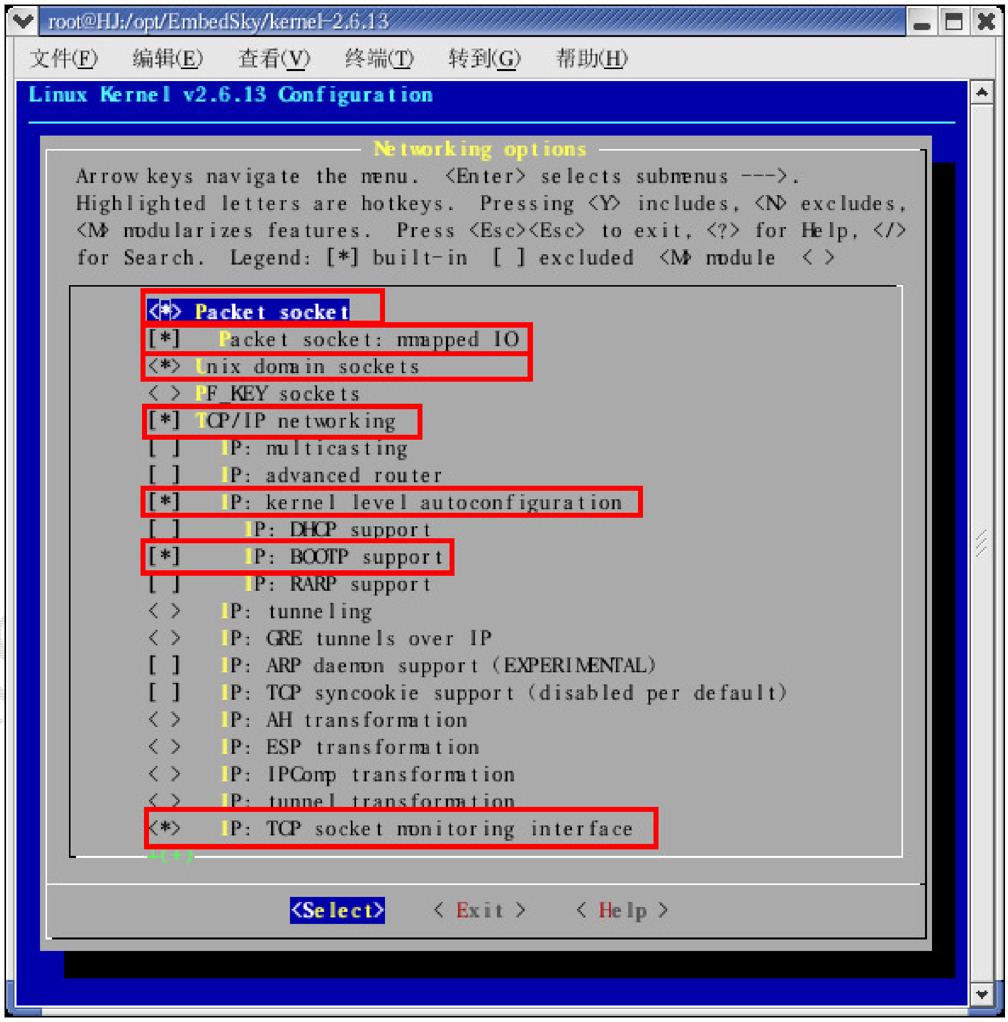
Select “Networking” in the main menu to configure the network protocol support:



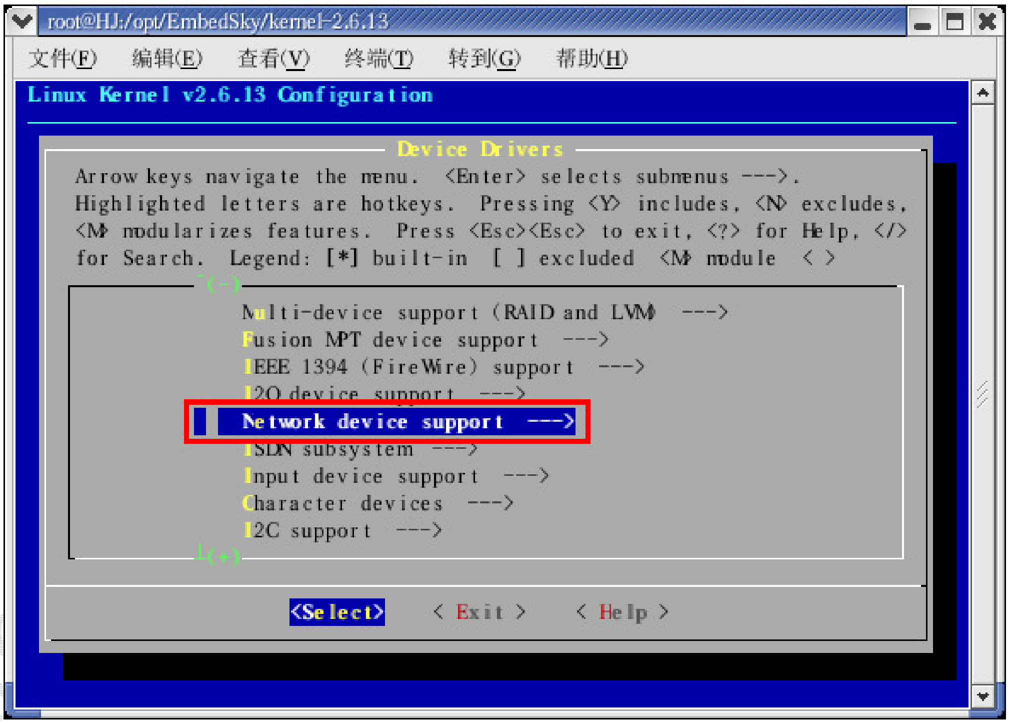
Select “Networking support” in the sub-menu “Networking” and get in:



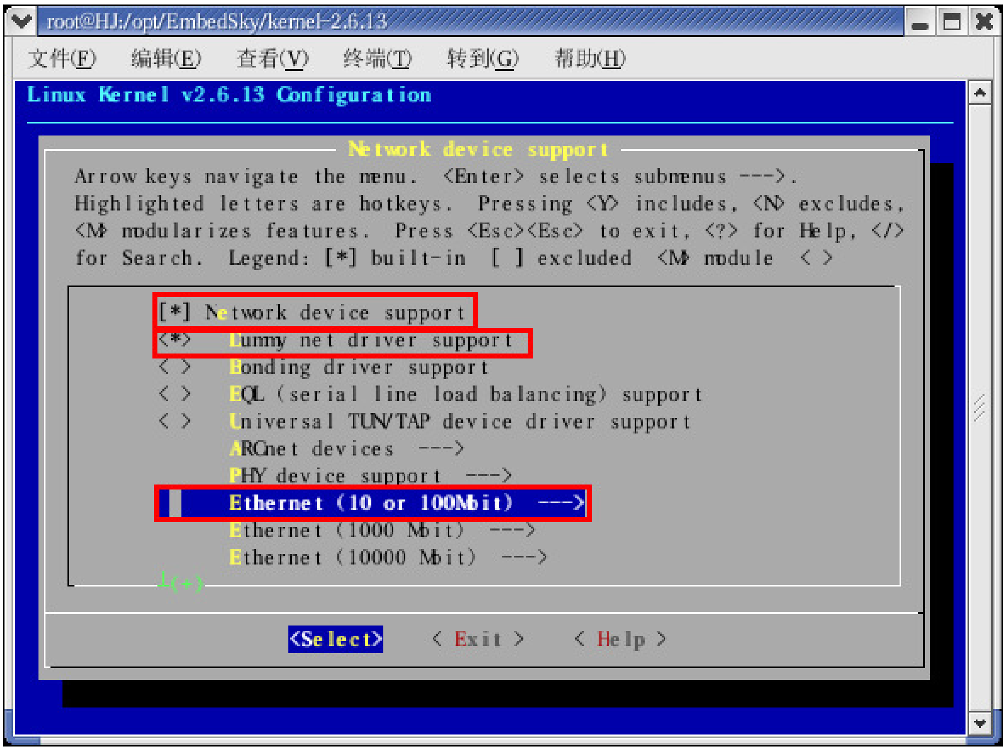
Select the following options in the sub-menu “Networking options”:



After the upper operation, go back to the main menu. And select “Network device support” in the menu “Device Drivers”:

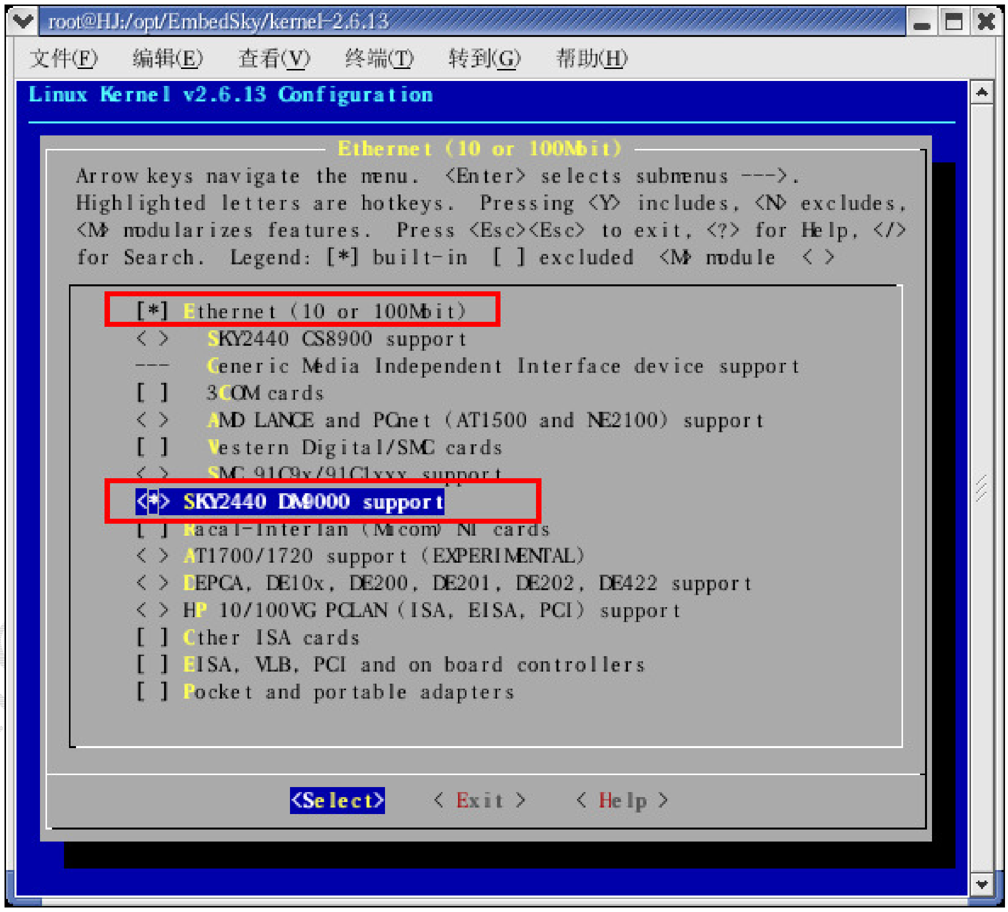


Select “Network device support” and “Dummy net driver support” in the sub-menu “Network device support”, and enter into the sub-menu “Ethernet (10 or 100Mbit)”:



Select “Ethernet (10 or 100Mbit)” in the sub-menu “Ethernet (10 or 100Mbit)”:

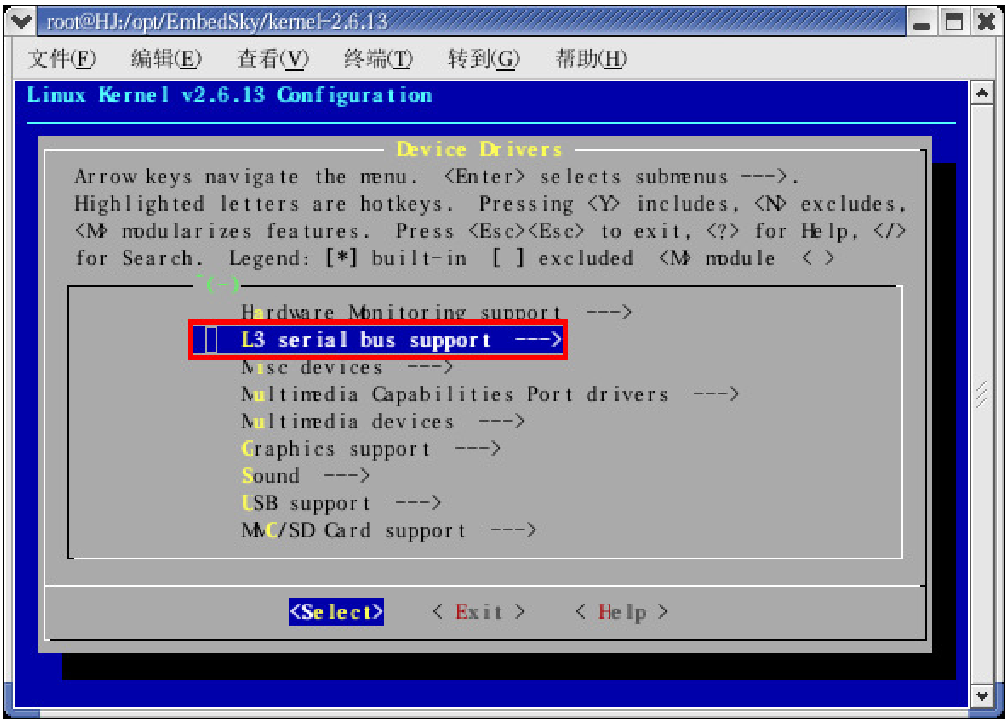
- Select “DM9000 support” to activate 100M network card:



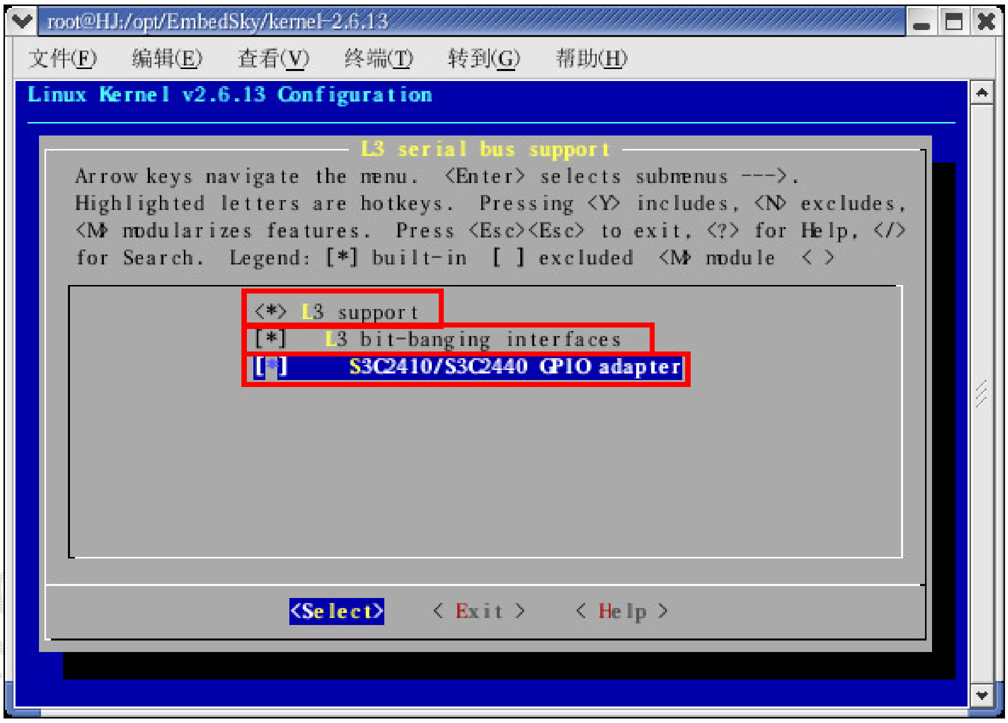
Select “<Exit>” twice and go back to the menu “Device Drivers”.

➢ **Configure sound card:**

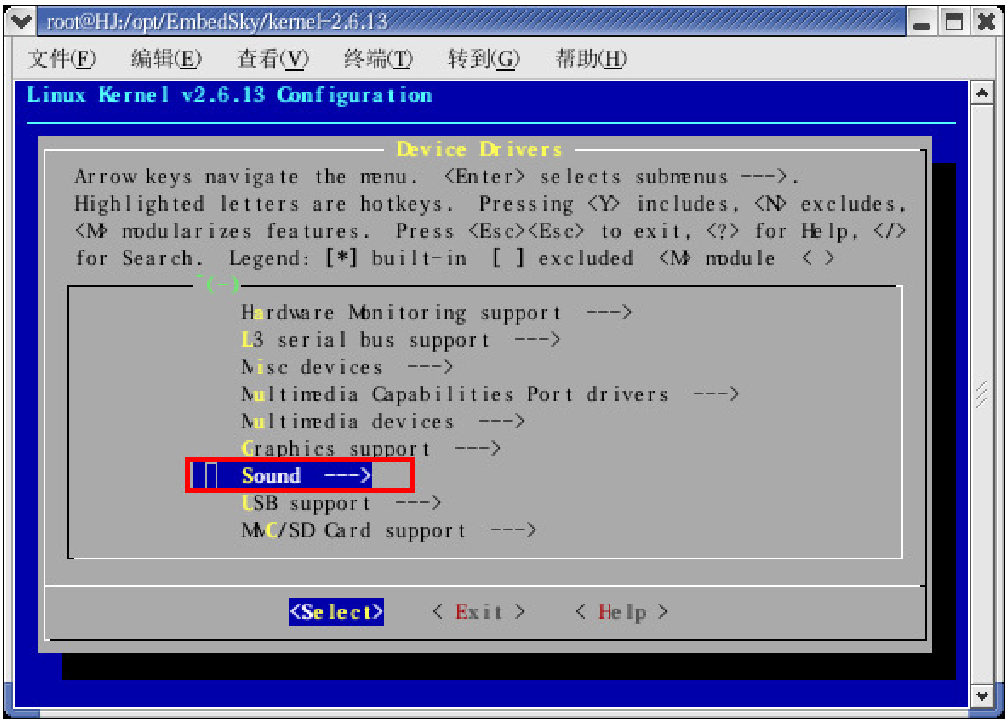
Select “L3 serial bus support” in the menu “Device Drivers” before configuring the sound card:



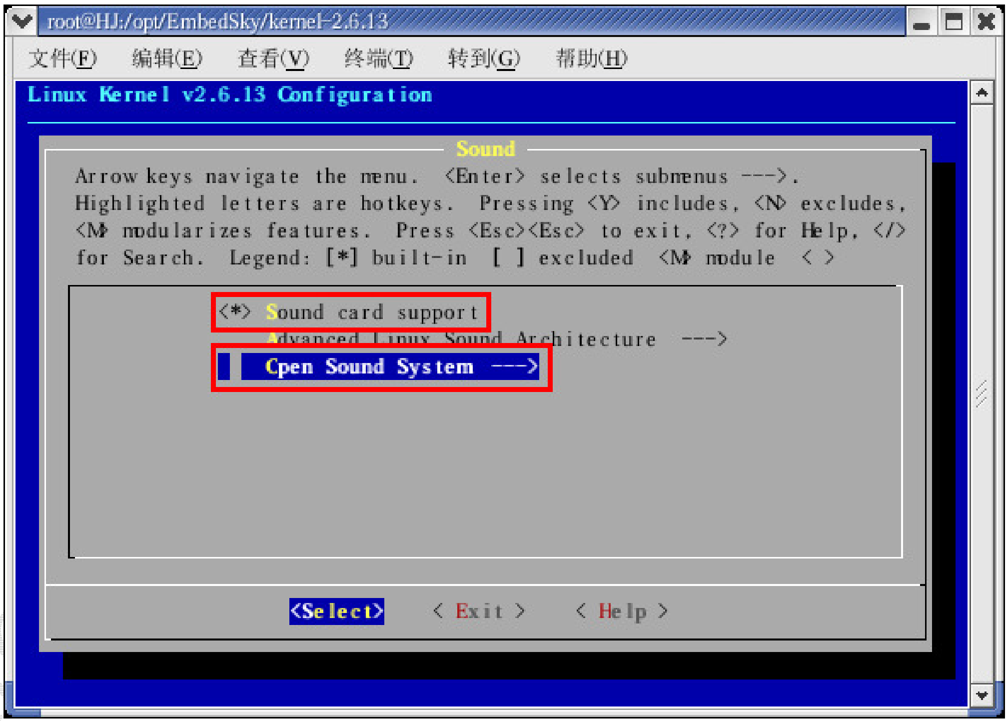
Select “L3 support”, “L3 bit-banging interfaces” and “S3C2410/S3C2440 GPIO adapter”:



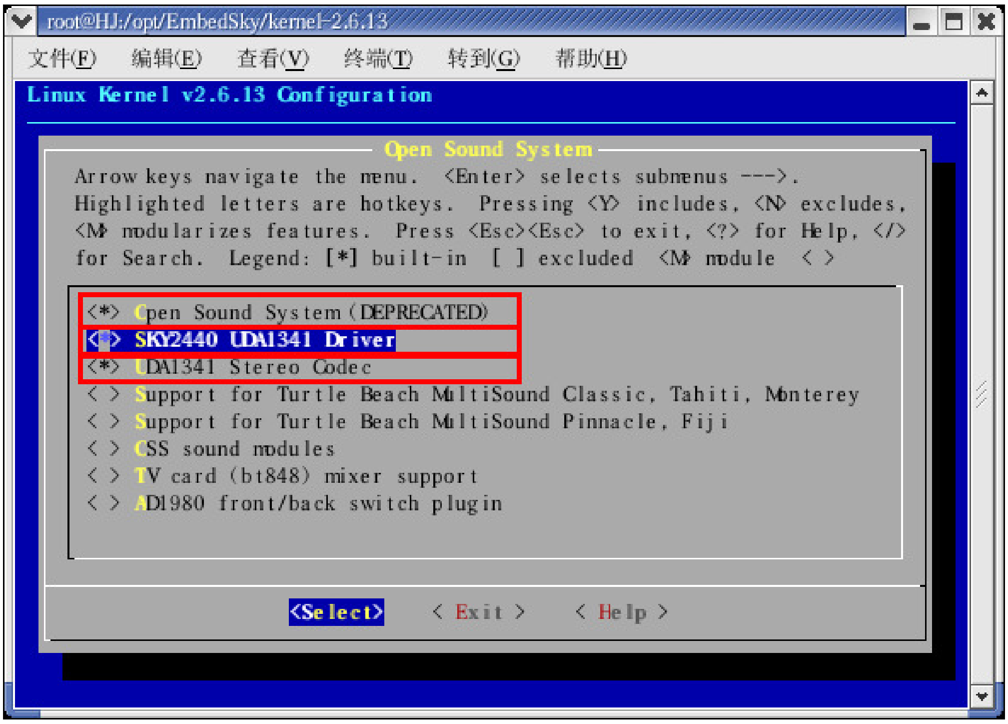
Select “<Exit>” to go back to the menu “Device Drivers”, and select “Sound”:



Select “Sound card support” and get into the sub-menu “Open Sound System”:



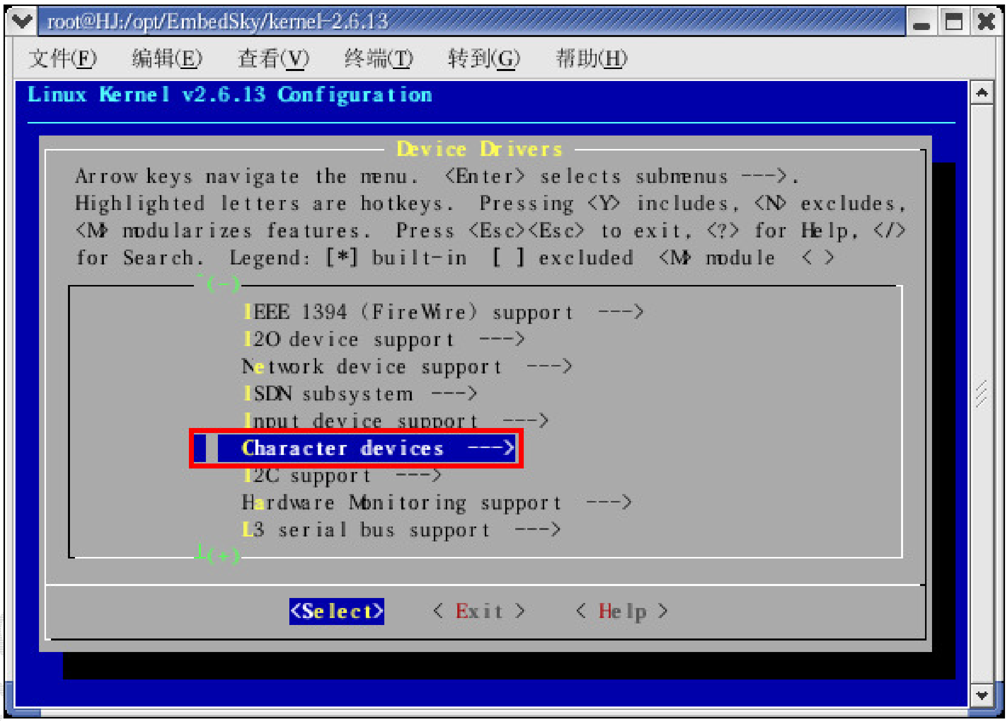
Select “Open Sound System (DEPRECATED)”, “TQ2440 UDA1341 Driver” and “UDA1341 Stereo Codec” in the menu “Open Sound System”:



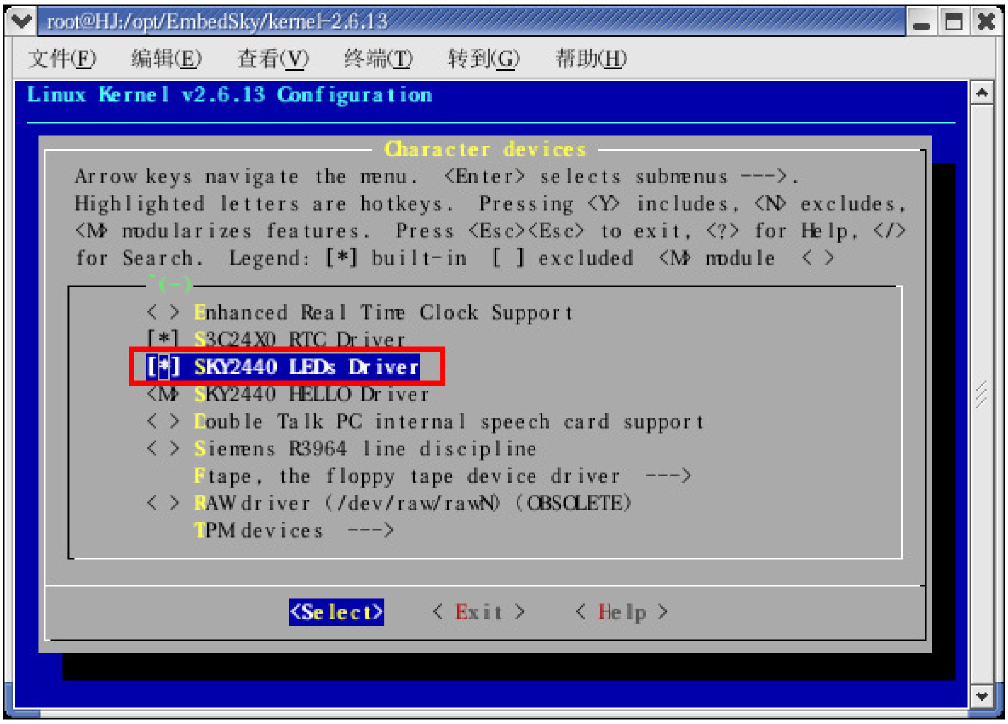
Select “<Exit>” twice to go back to the menu “Device Drivers”.

➢ **Configure user LED:**

Select “Character devices” in the menu “Device Drivers” and press return-key to get in:

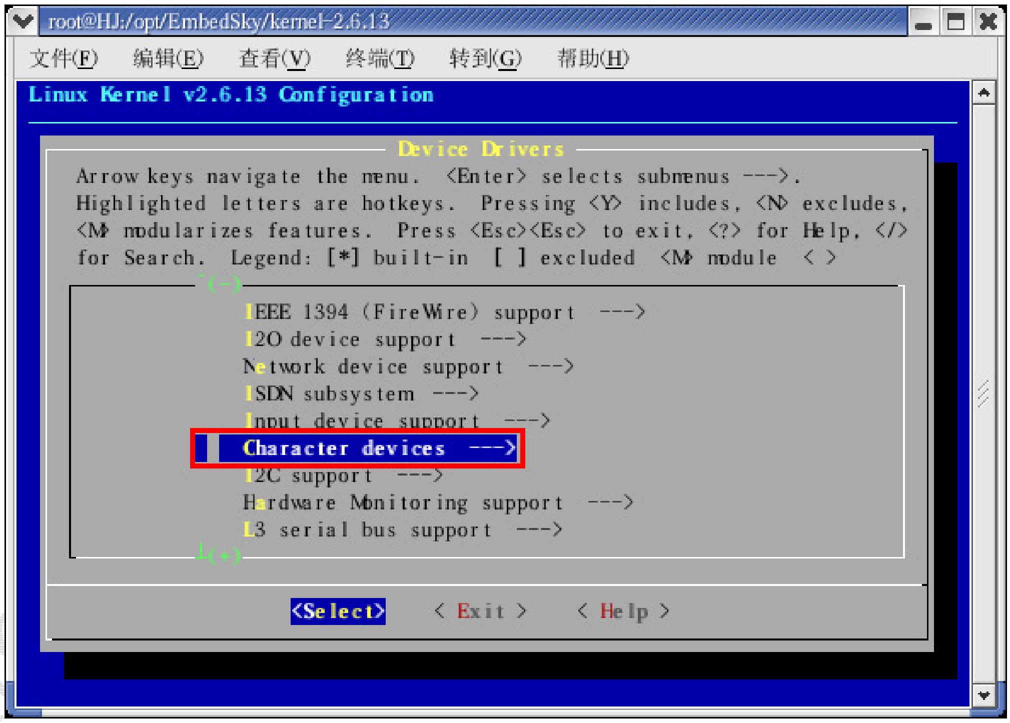


Select “TQ2440 LEDs Driver”:



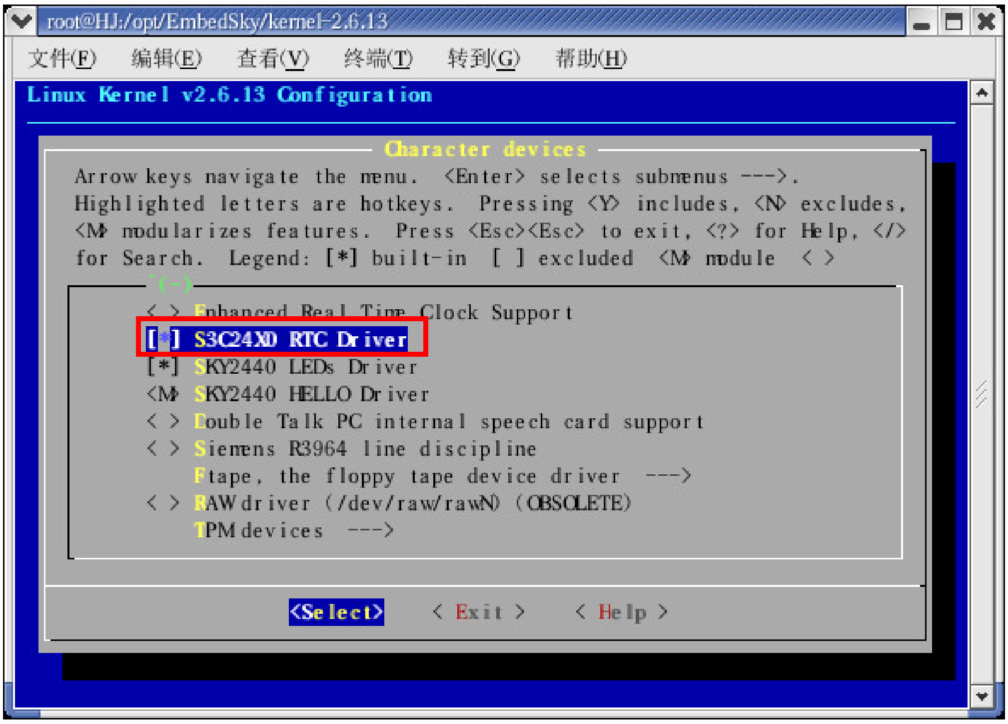
➢ **Configure real-time clock:**

Select “Character devices” in the menu “Device Drivers” and press return-key to get in:



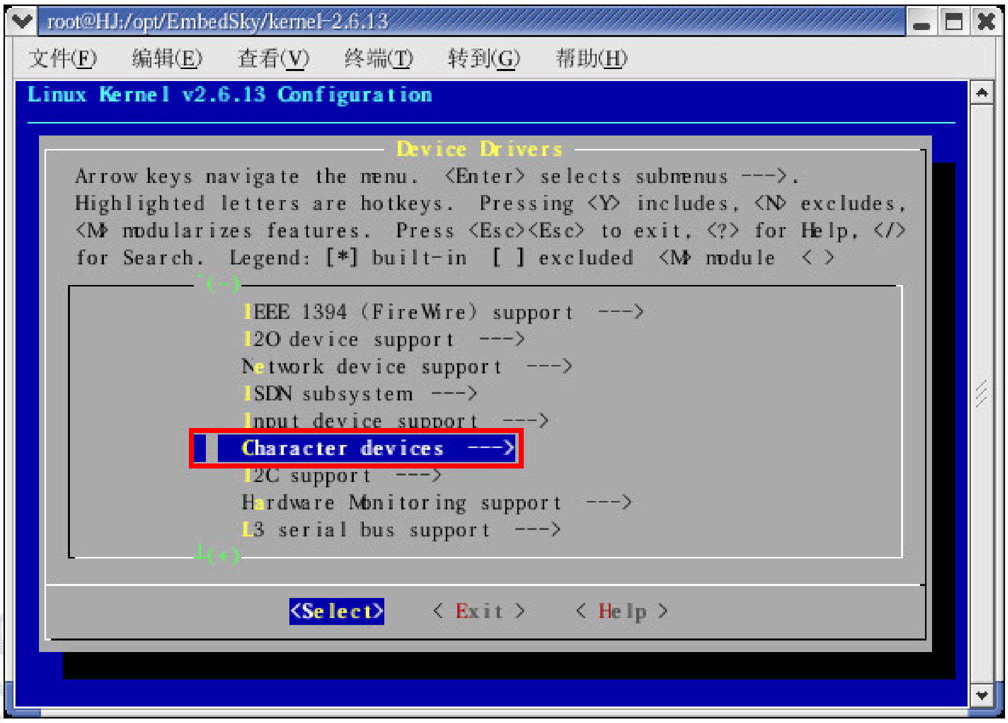
Select “S3C24X0 RTC Driver”:

Caution: Don’t select “Enhanced Real Time Clock Support”.

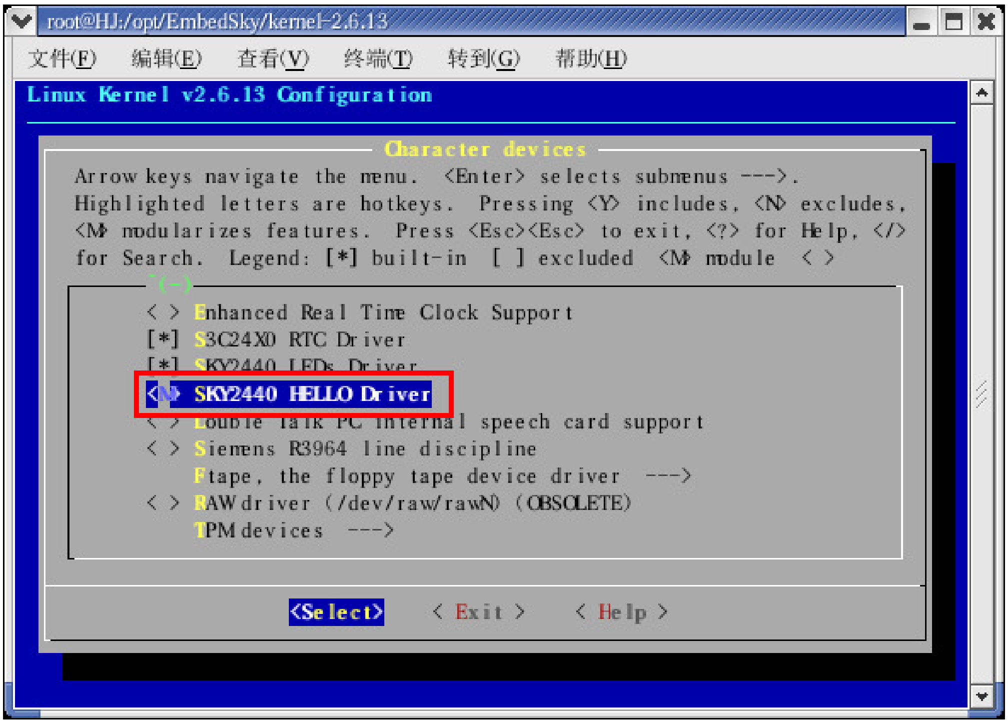


➢ **Configure TQ2440_Hello driver:**

Select “Character devices” in the menu “Device Drivers” and press return-key to get in:

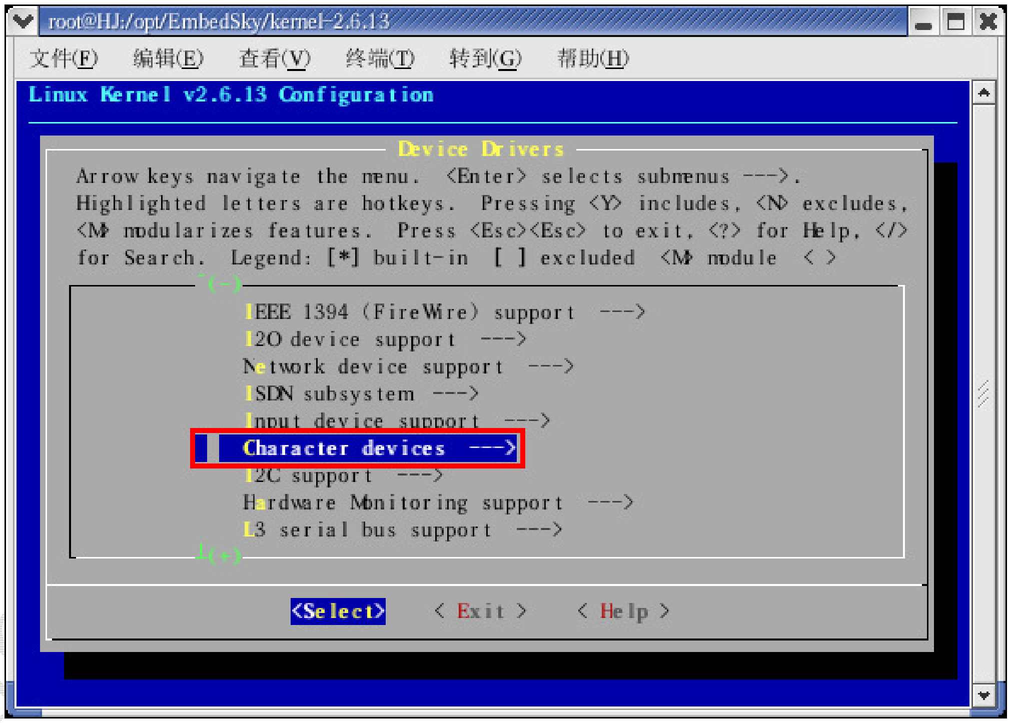


Set “M” in the option “TQ2440 HELLO Driver”:

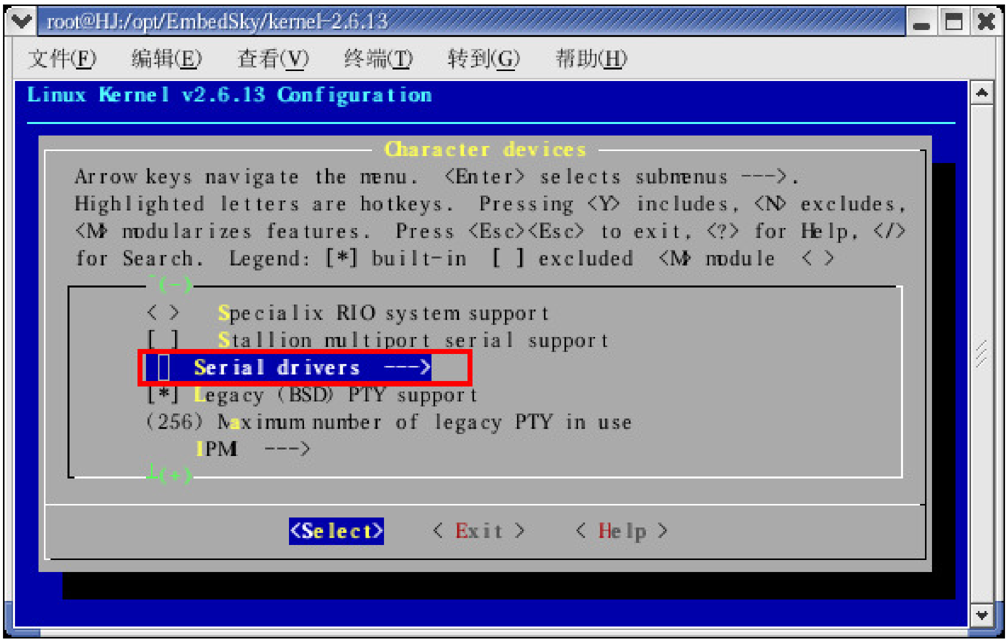


➢ **Configure serial port:**

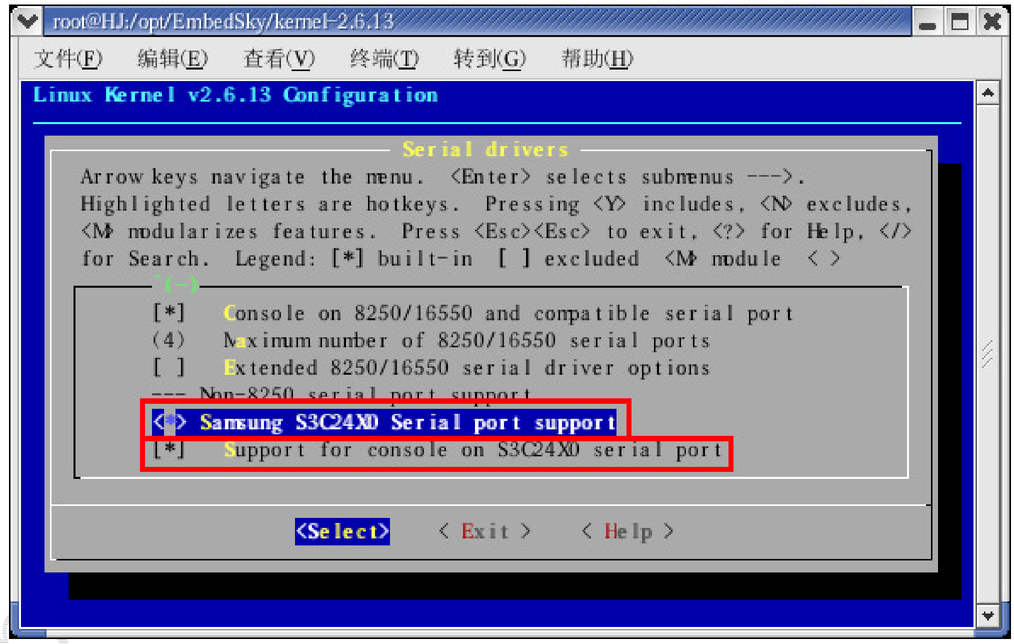
Select “Character devices” in the menu “Device Drivers” and press return-key to get in:



Get into the sub-menu “Serial drivers”:



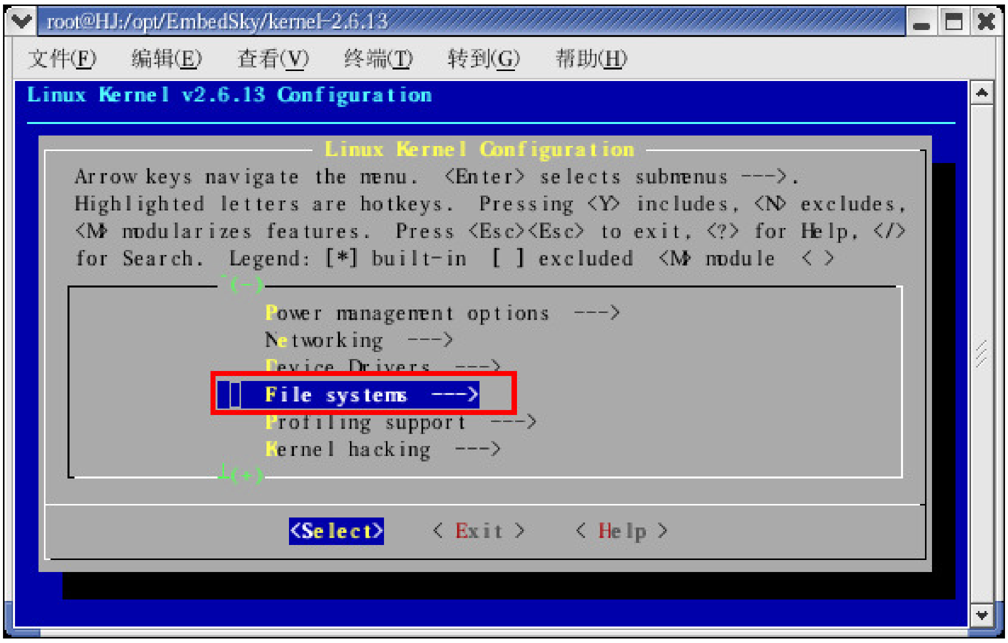
Select the following options:



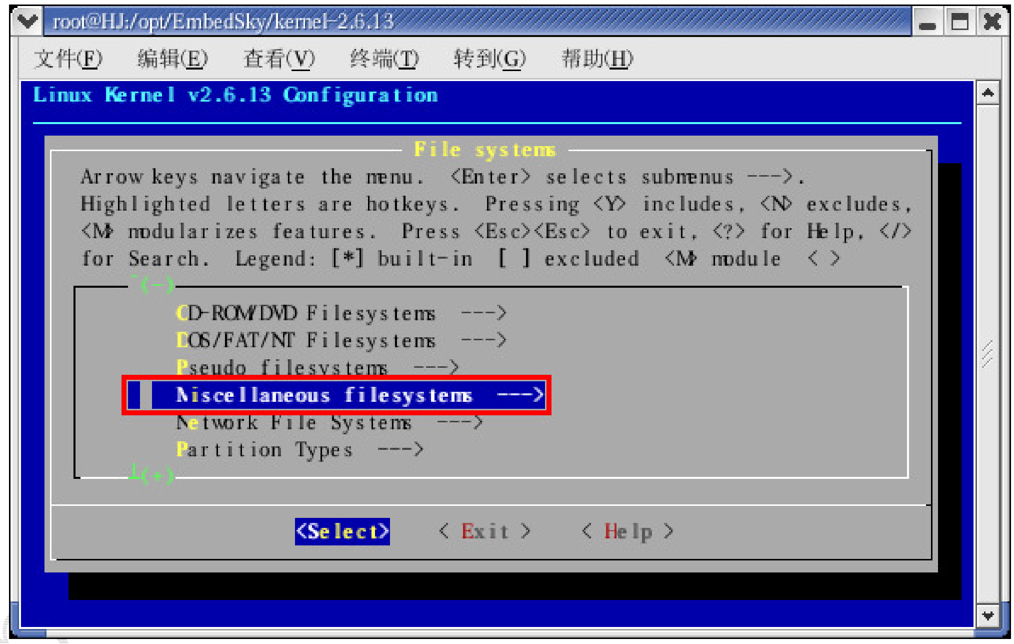
Select “<Exit>” twice to go back to the menu “Device Drivers”.

➢ **Configure Yaffs file system:**

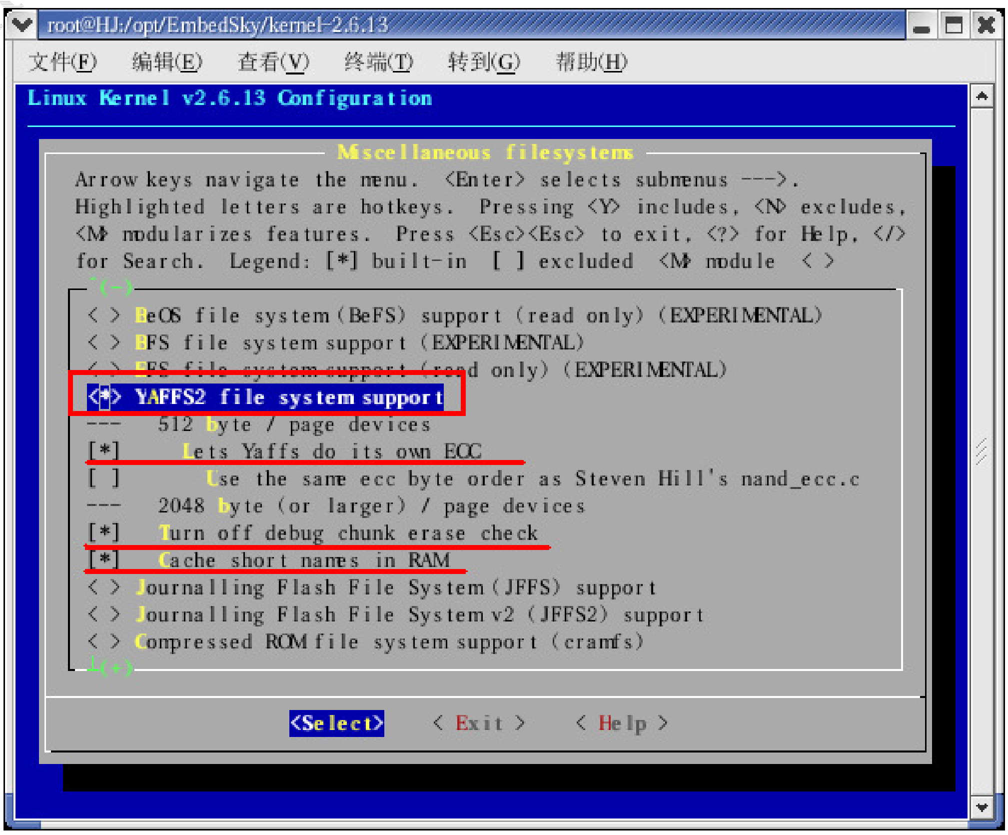
Select “File Systems” in the main menu and get in:



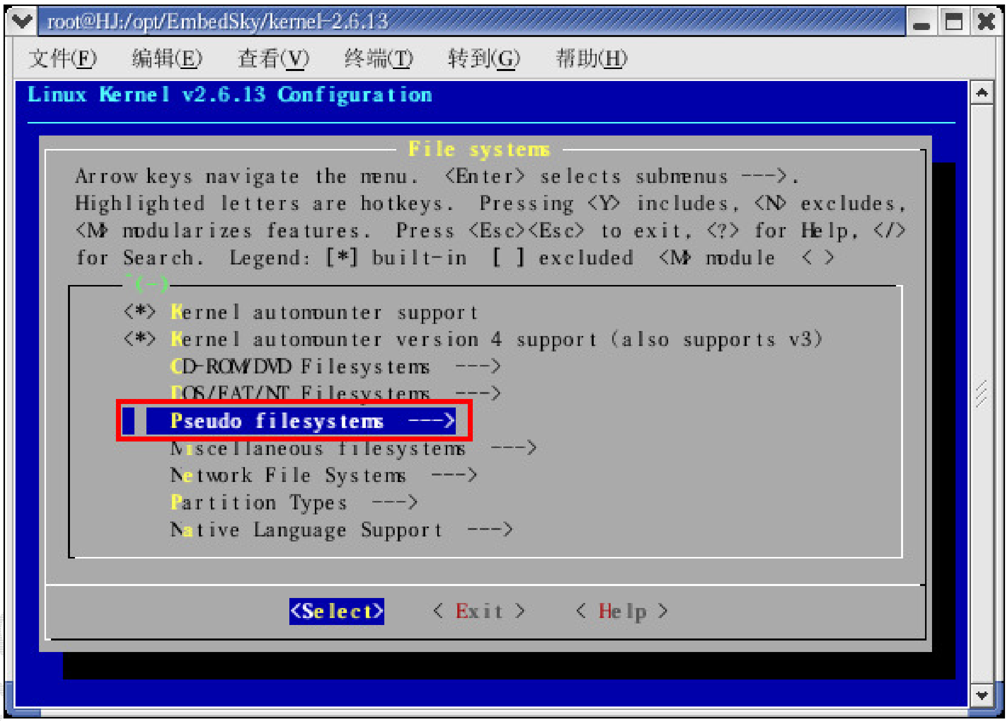
Select “Miscellaneous filesystems” in the menu “File Systems” and get in:



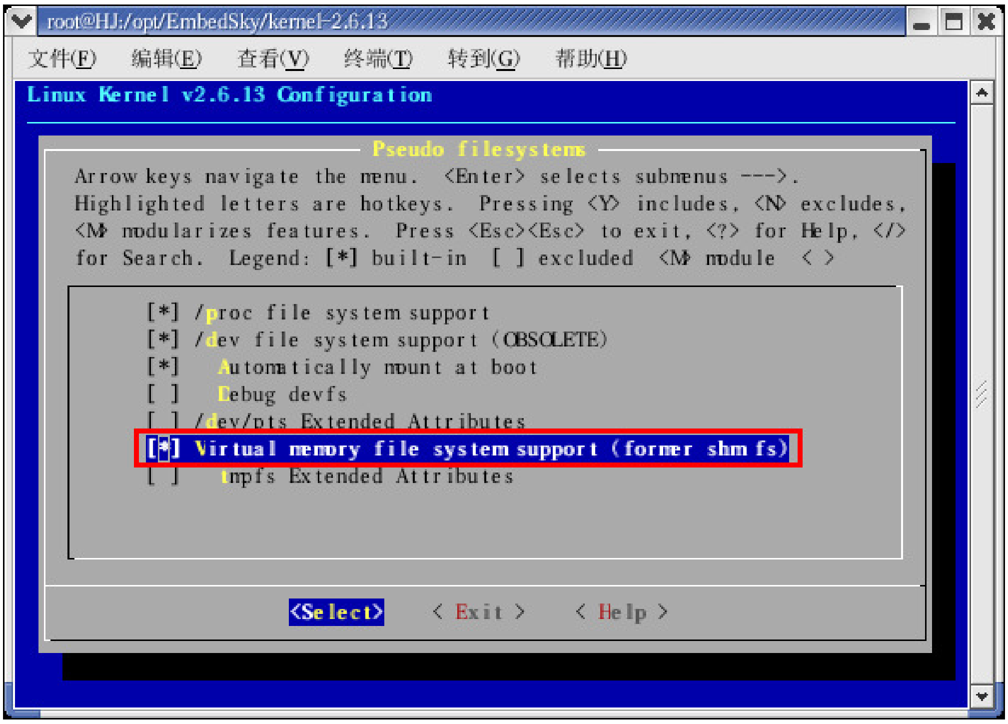
Select the following options:



Caution: Select "Pseudo filesystems" to make sure Yaffs file system works correctly:

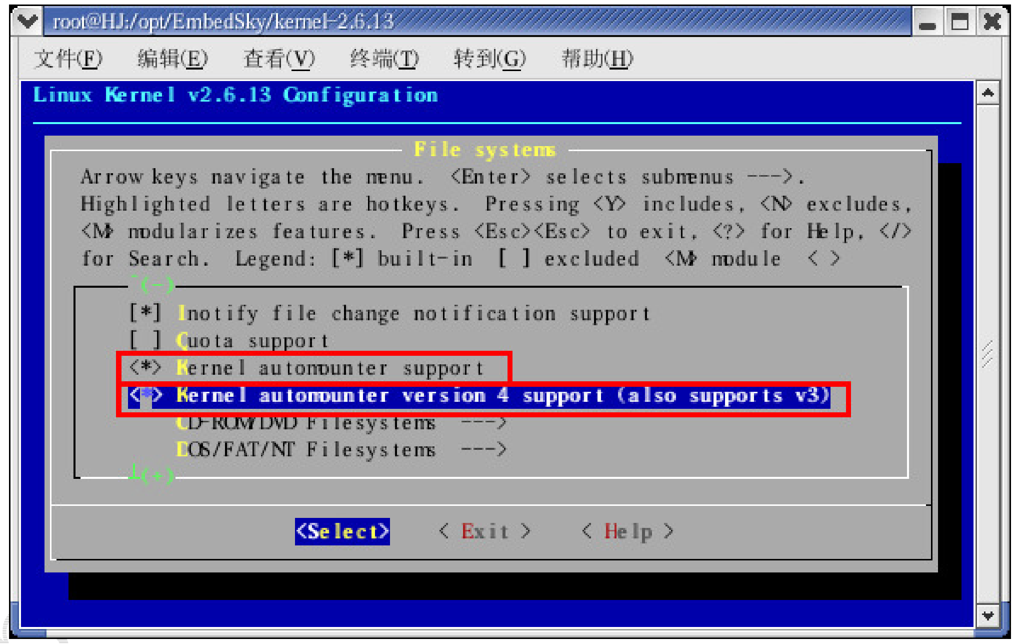


Select “Virtual memory file system support (former shm fs)”:



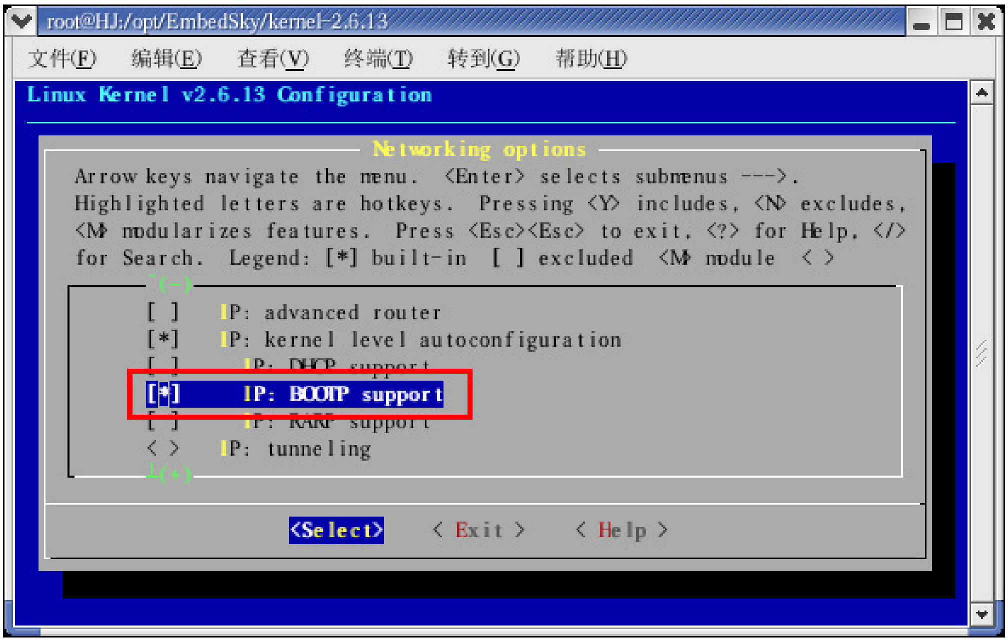
Select “<Exit>” to go back to the sub-menu “File System”.

Select “Kernel automounter support” and “Kernel automounter version 4 support (also supports v3)” to make sure the kernel supports automounter:

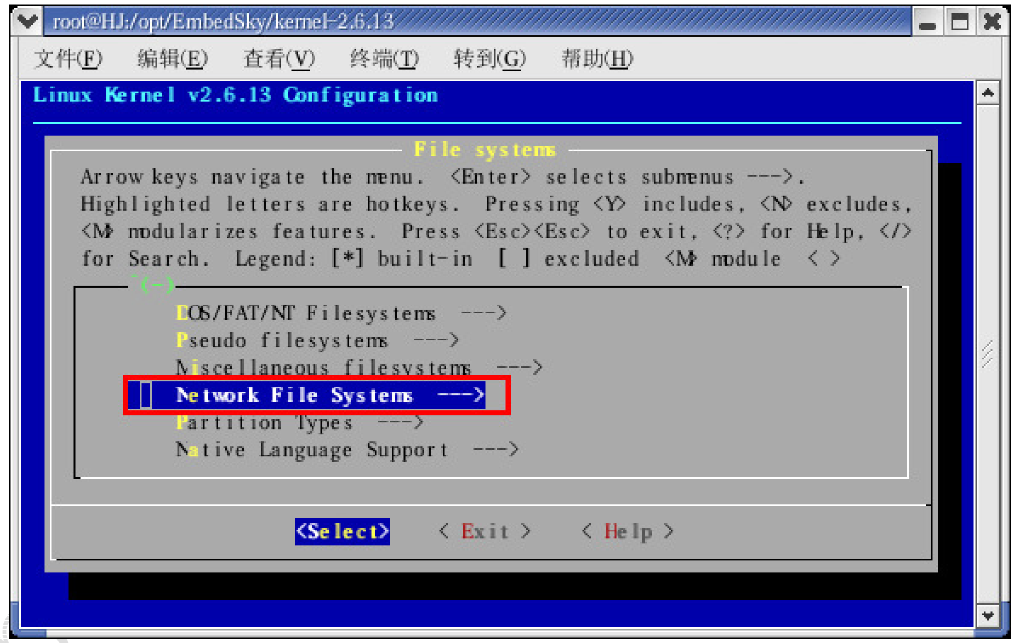


➢ **Configure NFS file system:**

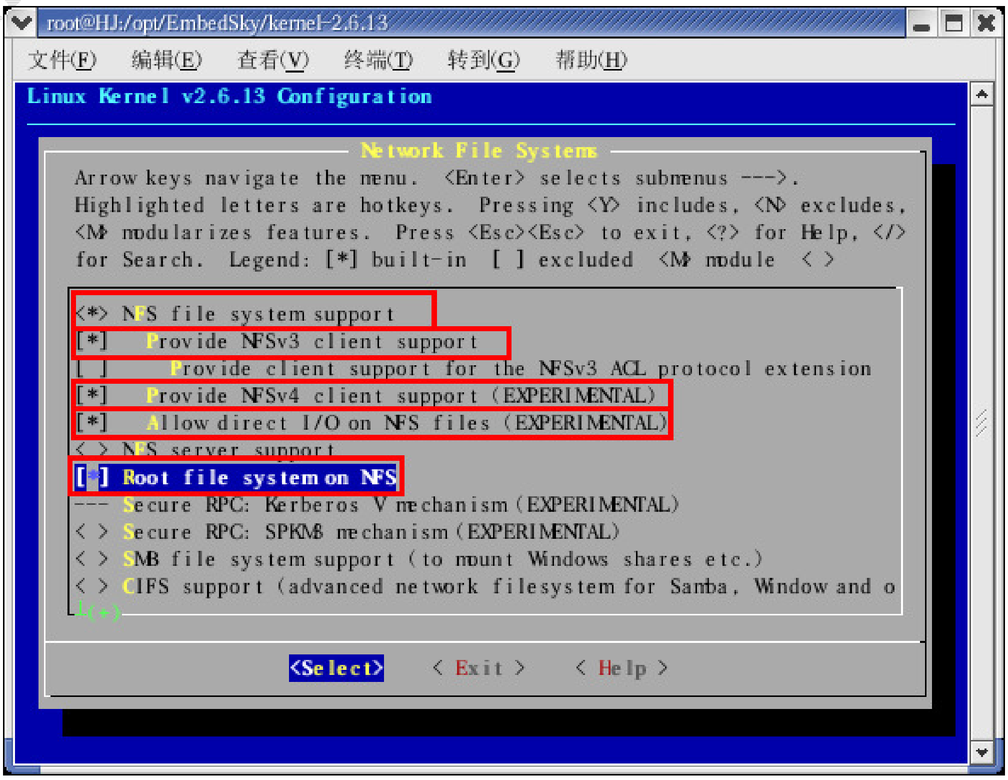
The option “IP: BOOTP support” has been selected in the previous network device configuration:



Select “Network File System” in the menu “File System” and get in:



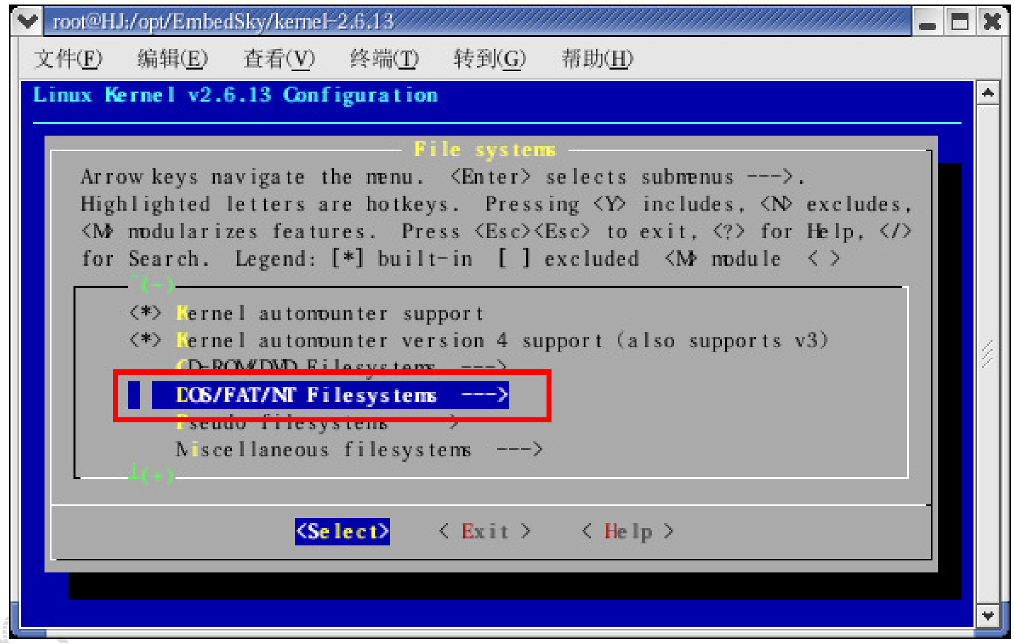
Select the following options:



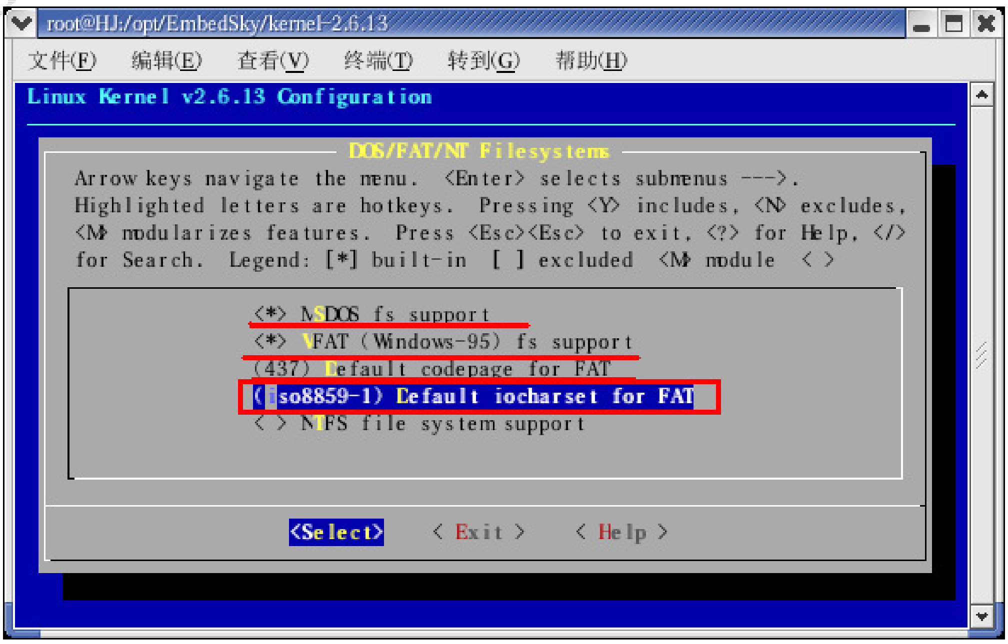
Select “<Exit>” to go back to the menu “File Systems”.

➢ **Configure the other file systems:**

Select “DOS/FAT/NT Filesystems” in the menu “File System” and get in:

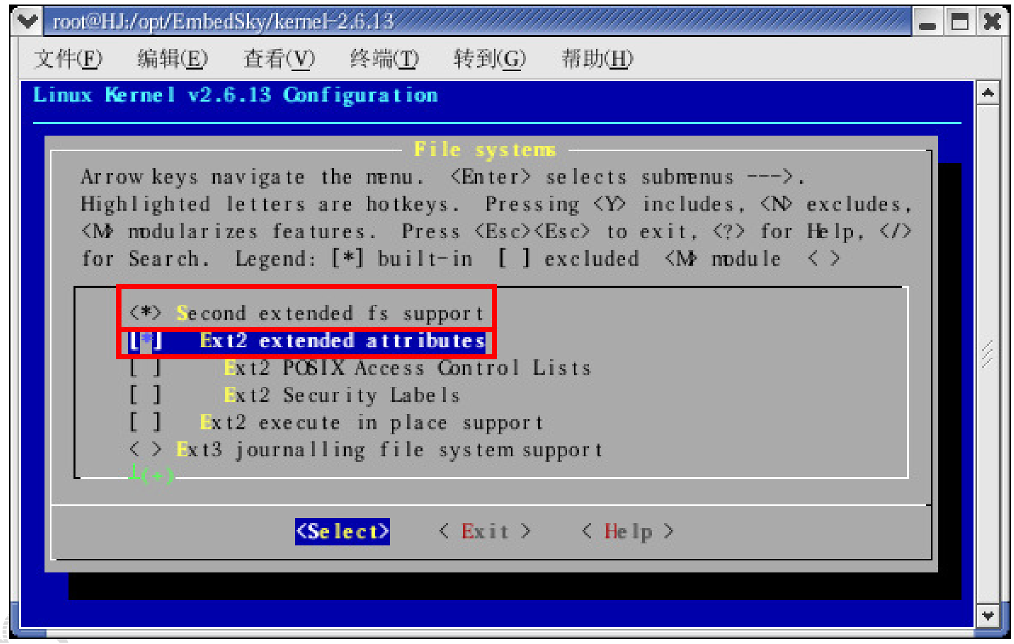


Select the following options:



Select “<Exit>” to go back to the menu “File Systems”.

Select “Ext2 extended attributes” to support EXT2 file system:



Most information about kernel configuration has been introduced in the upper contents. However, a more profound understanding towards kernel configuration needs more experiences.

4. 3 Making root file system

The root file system is loaded by Linux when initializing. The user can execute the command “root=” to set the device that is corresponding to the root file system. The following contents introduce the process building Linux file system:

4. 3. 1 Components of root file system

Linux root file system usually contains the following directories:

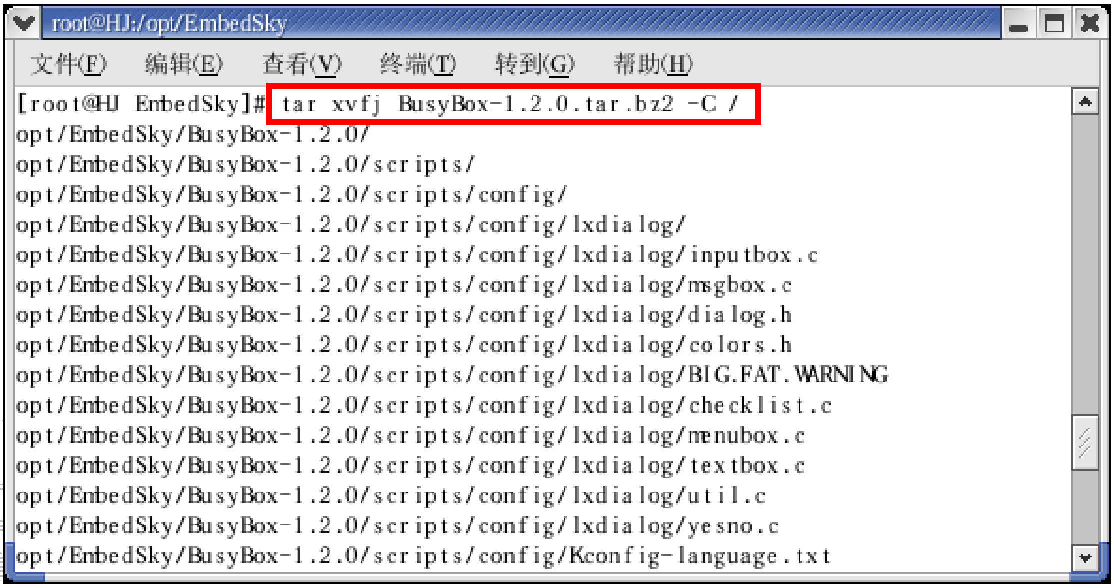
- /dev—the directory of device file node. Mount the device file in this directory.
- /proc—mount proc file system in this directory.
- /bin—basic storage system commands are in this directory.
- /etc—system start-up configuration scripts, like rcS, inittab, fstab and so on are in this directory.
- /lib—directory of system default dynamic link liberary.
- /usr—user directory, includes “/usr/bin”, “/usr/sbin” and so on.
- /sbin—basic storage system commands are in this directory.
- /tmp—temporary directory, it is not necessary.
- /var—this directory contains general variables used by system, and the size of it usually changes.

4. 3. 2 BusyBox compiling

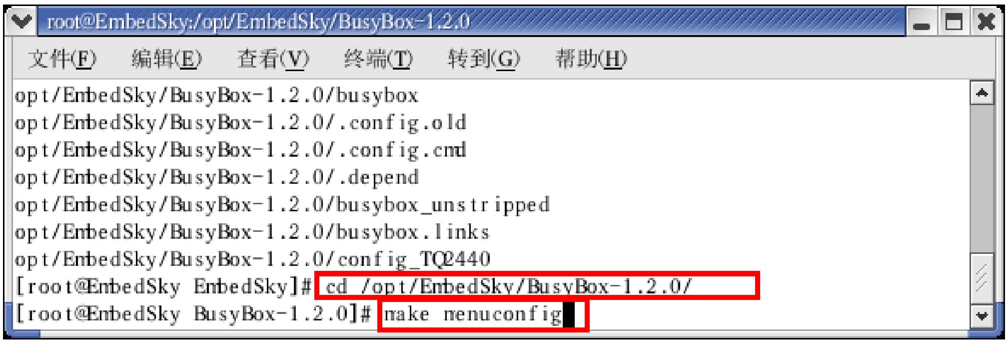
BusyBox is a UNIX system tool set which contains most of the ordinary commands. The service of BusyBox is utilized by executing link instructions.

The following steps illustrate how to compile BusyBox:

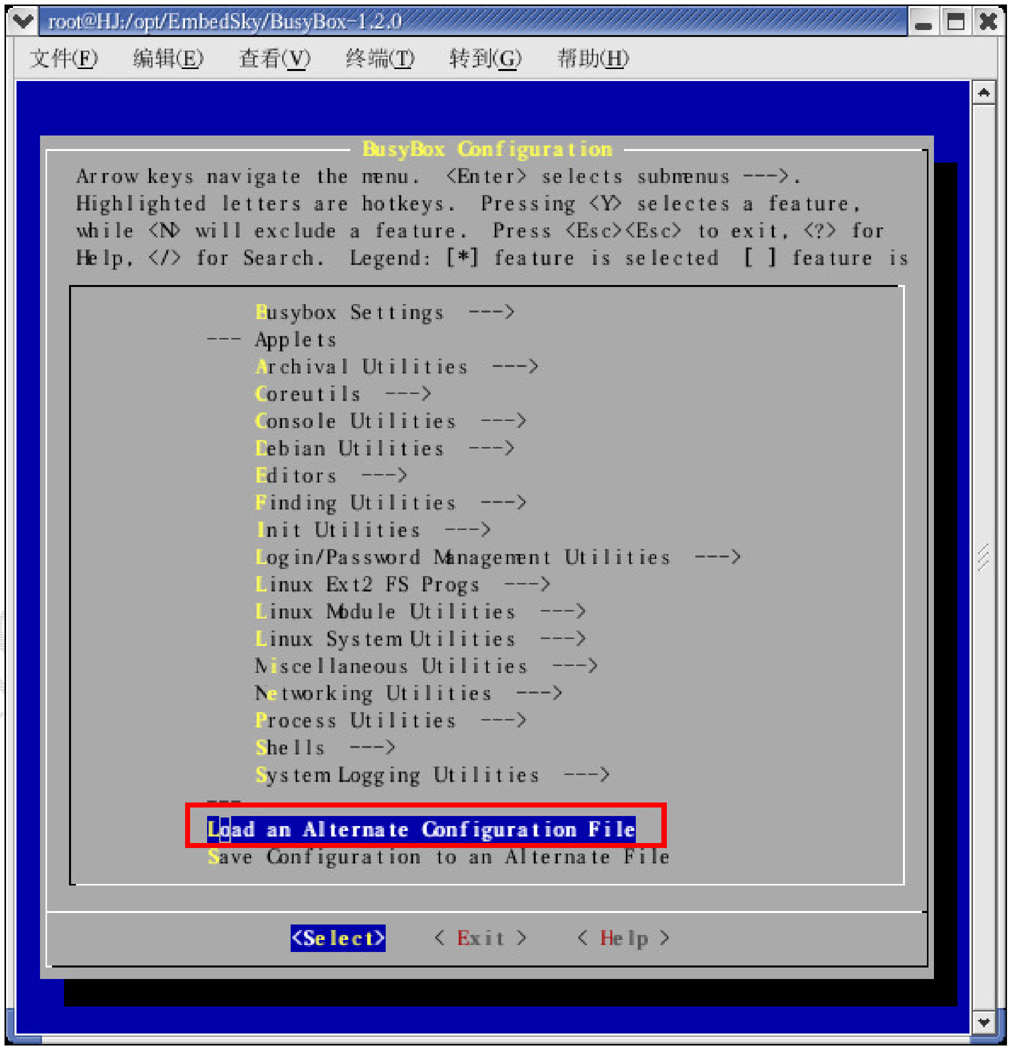
Step1, decompress BusyBox. The source code of BusyBox is the file “BusyBox.tar.bz2” under the directory “Linux”. Copy the file to the directory “/opt/EmbedSky/” in PC and execute the command “tar xvfj BusyBox.tar.bz2 -C /” to decompress it:



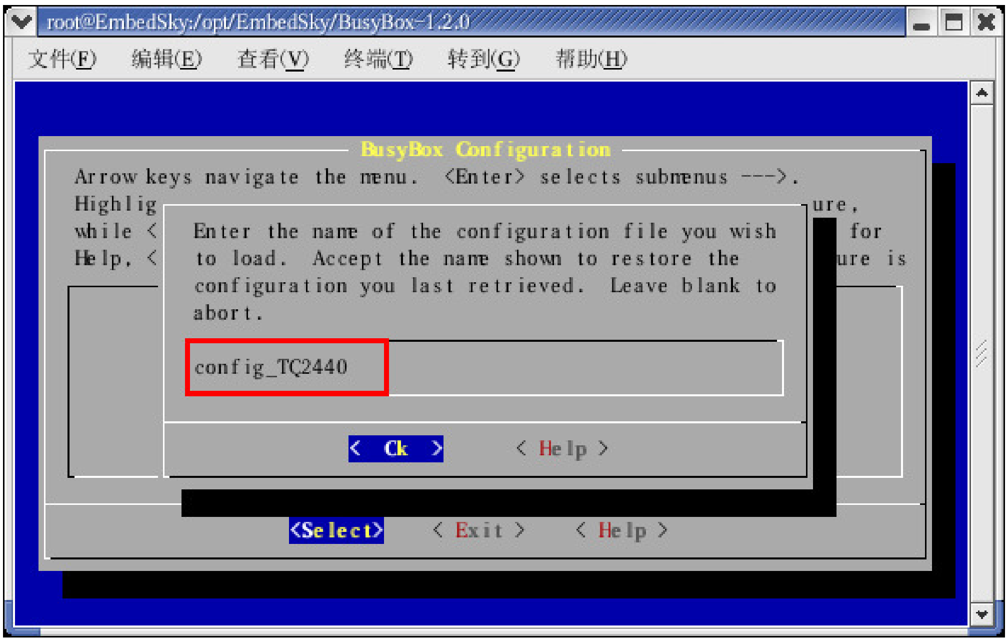
Step2, get into the directory “/opt/EmbedSky/BusyBox-1.2.0/” and execute the command “make menuconfig”:



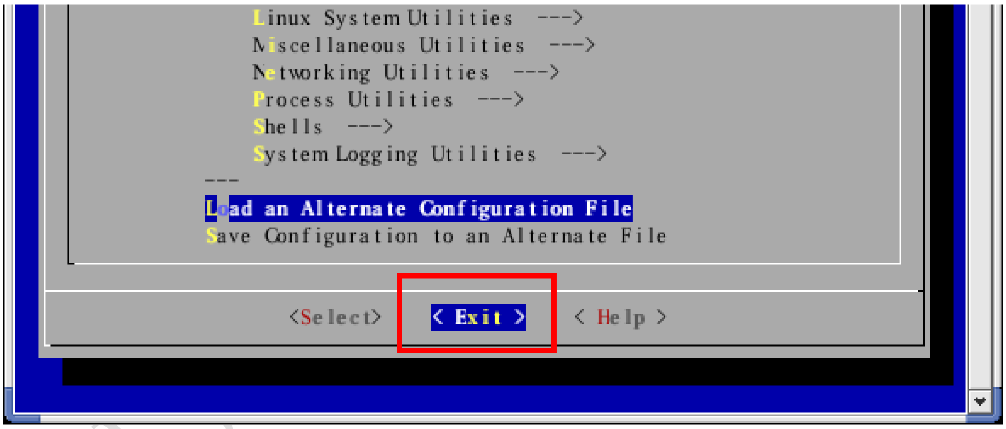
Step3, select “Load an Alternat Configuration File” in the following diagram:



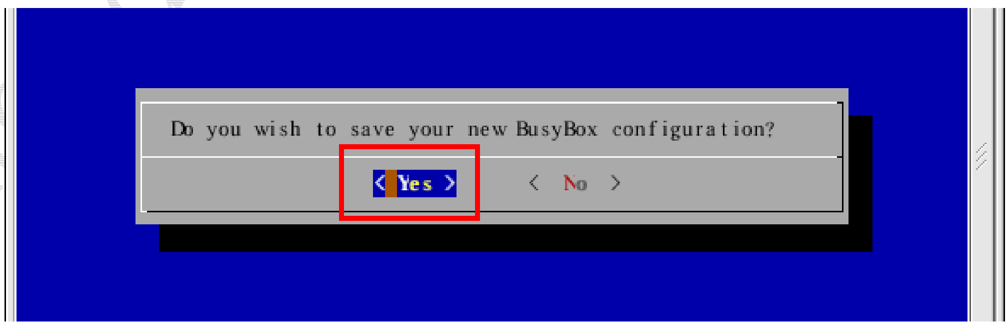
Step4, enter the name of configuration file “config_TQ2440” and press return-key to continue:



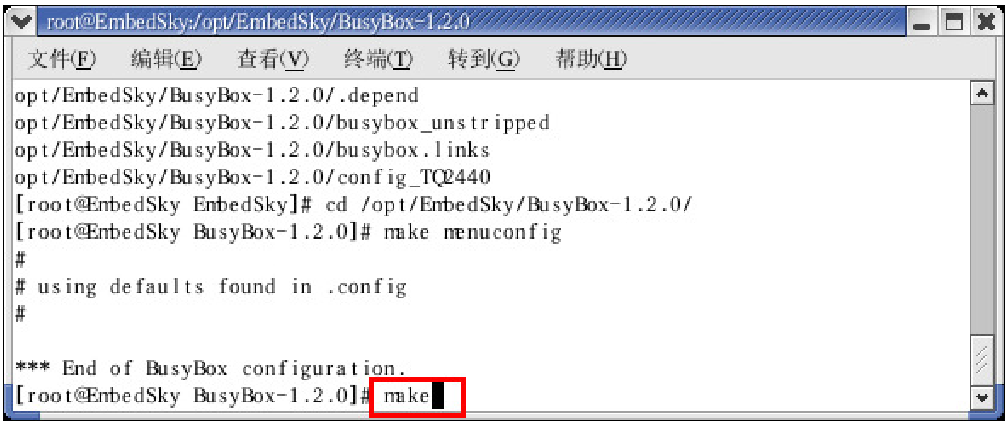
Step5, select “<Exit>” and press return-key to continue:



Select “<Yes>” to save the configuration:

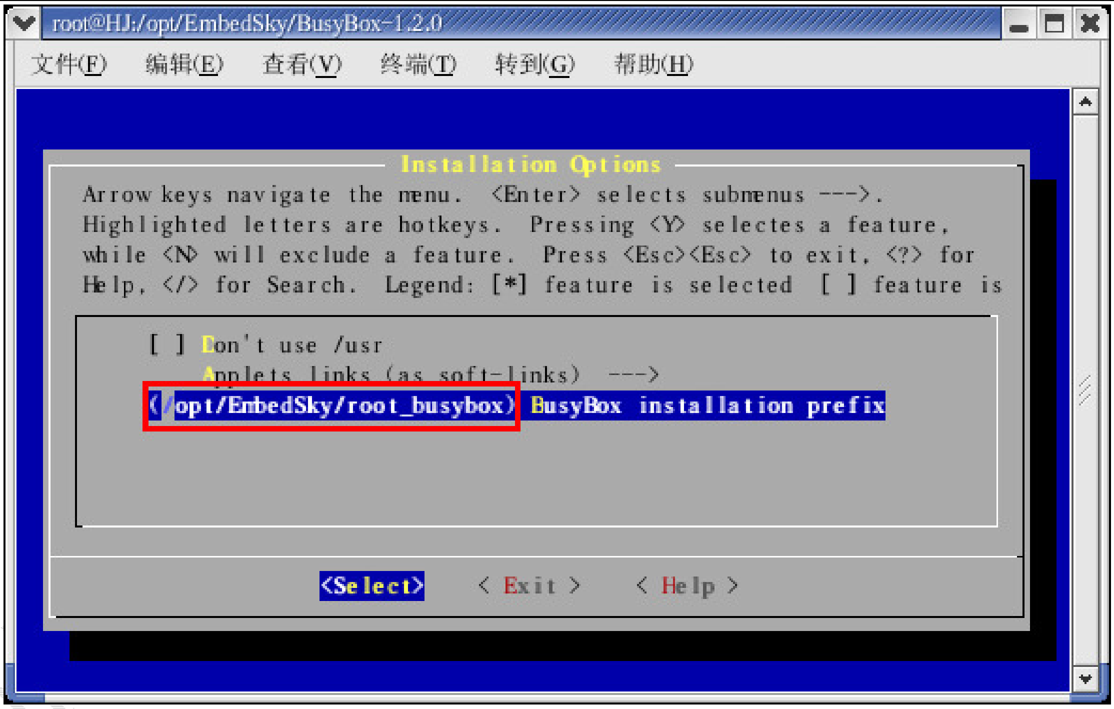


Step6, execute the command “make” to start compiling BusyBox:

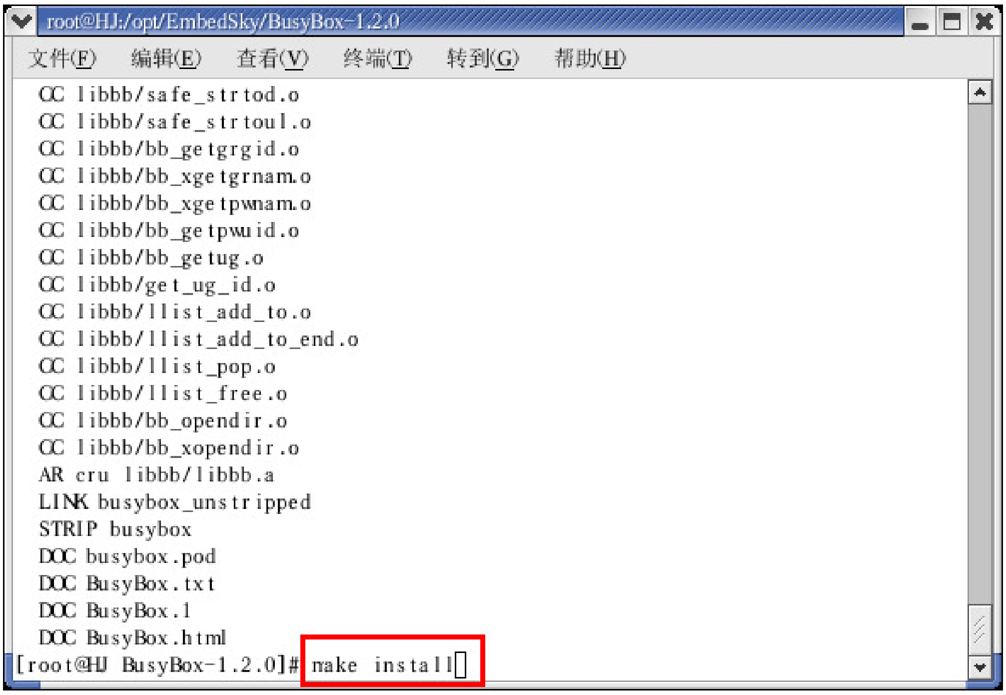


Step7, intall BusyBox after compiling. The installation path is “/opt/EmbedSky/root_busybox/”:

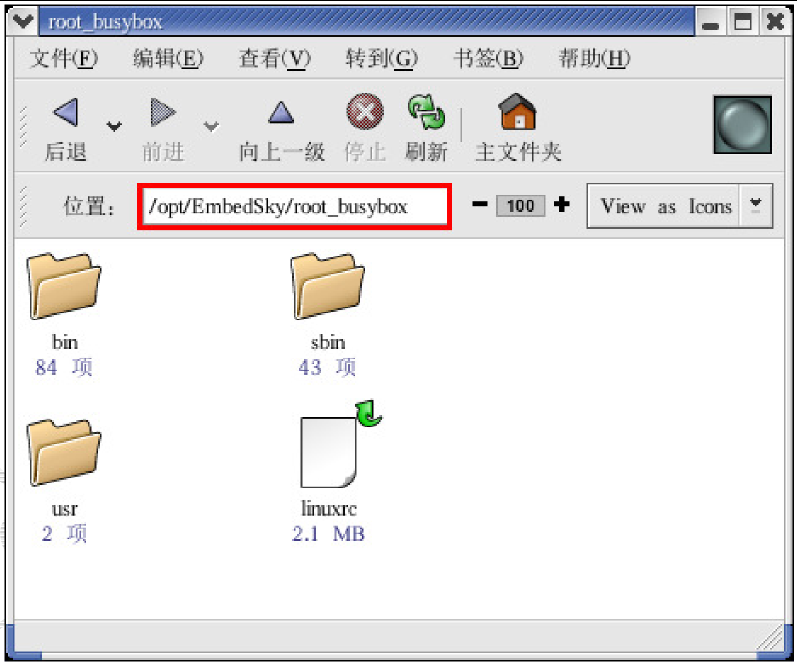
Set the installation path: “Busybox Settiings” -> “Installation Options” -> “BusyBox Installation prefix”



Execute the command "make install" to start installation:



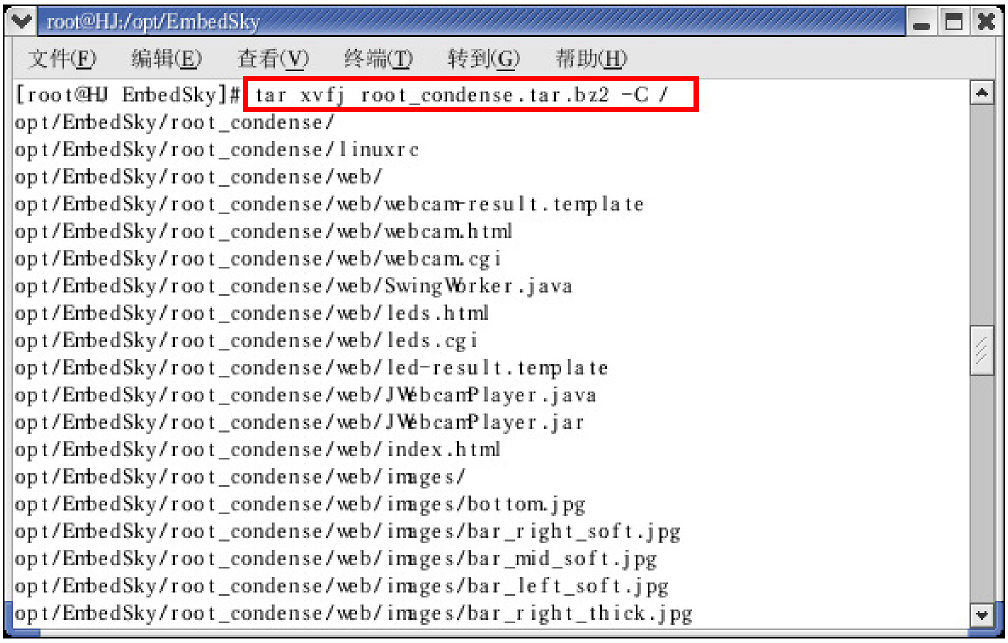
Step8, the 4 directories "bin/", "sbin/", "usr/bin/" and "usr/sbin/" and the file "linuxrc" appear under the directory "/opt/EmbedSky/root_busybox/" after installation: ("linuxrc" is actually the shortcut of the file "bin/busybox")



After compiling, add all the files mentioned in “3.3.1 节” to the directory “root_busybox”. And then the BusyBox file system is built completely.

4. 3. 3 Building root file system

Decompress “root_condense.tar.bz2” under the directory “Linux” to “/opt/EmbedSky”. Execute the command “tar xvfj root_condense.tar.bz2 -C /”:



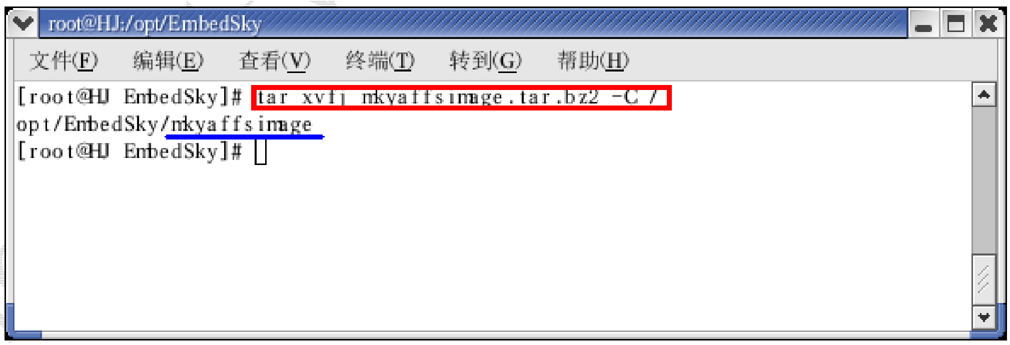
Copy the BusyBox compiled in “4.3.2 节” to the corresponding directory under “root_condense”.

4. 3. 4 Making Yaffs root file system image

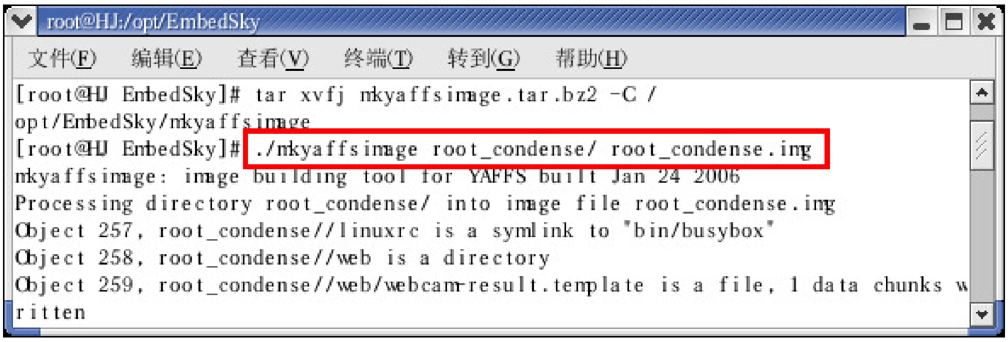
The software “mkyaffsimage” is needed when making Yaffs file system image.

“mkyaffsimage” is in the compression package “mkyaffsimage.tar.bz2” and “crosstools_all.tar.bz2” under the directory “Linux” in CD-ROM. “mkyaffsimage” has been installed previously when installing cross-compiler. The user can re-install it here. Decompress the file “mkyaffsimage.tar.bz2”, and find “mkyaffsimage”:

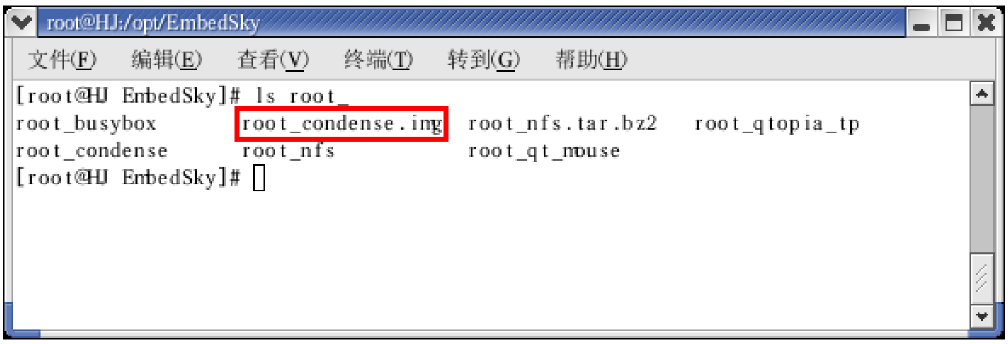
(the decompression command is shown in the red frame, and the executable program decompressed which is under the directory “/opt/EmbedSky/” is highlighted by the blue underline.)



Execute the command “./mkyaffsimage root_condense/ root_condense.img” under the directory “/opt/EmbedSky/” to make Yaffs root file system image.



The red frame marks the created file:



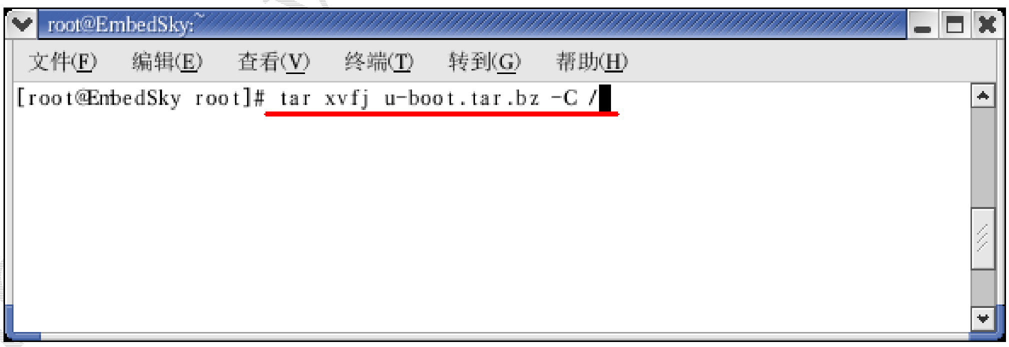
Follow the steps of burning file system introduced previously to burn “root_condense.img” to platform.

4.4 Compiling u-boot

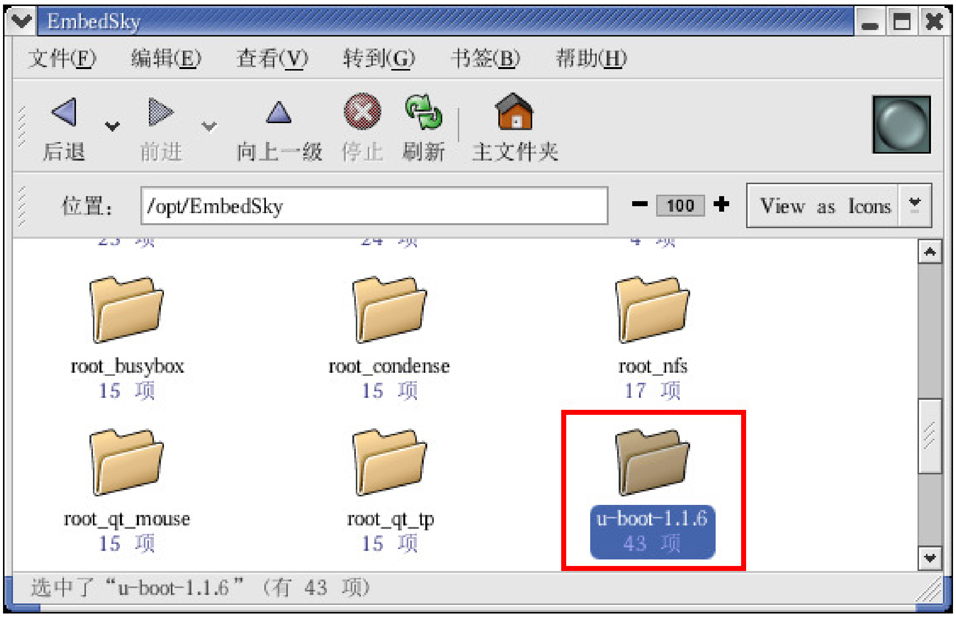
The file “u-boot.tar.bz2” in CD-ROM is the source code package of u-boot for TQ2440.

4.4.1 Decompressing u-boot

Execute the decompression command: tar xvfj u-boot.Tar.bz2 -C /



The file is auto-decompressed to the directory “/opt/EmbedSky/u-boot-1.1.6/”:



4.4.2 Configuring u-boot

Execute the command: make TQ2440_config

```
[root@EmbedSky EmbedSky]# cd /opt/EmbedSky/u-boot-1.1.6/
[root@EmbedSky u-boot-1.1.6]# make SKY2440_config
Configuring for SKY2440 board...
[root@EmbedSky u-boot-1.1.6]#
```

3.4.3 Compiling u-boot

Execute the command: make

```
[root@EmbedSky EmbedSky]# cd /opt/EmbedSky/u-boot-1.1.6/
[root@EmbedSky u-boot-1.1.6]# make SKY2440_config
Configuring for SKY2440 board...
[root@EmbedSky u-boot-1.1.6]# make
for dir in tools examples post post/cpu ; do make -C $dir _depend ; done
make[1]: Entering directory `/opt/EmbedSky/u-boot-1.1.6/tools'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/opt/EmbedSky/u-boot-1.1.6/tools'
make[1]: Entering directory `/opt/EmbedSky/u-boot-1.1.6/examples'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/opt/EmbedSky/u-boot-1.1.6/examples'
make[1]: Entering directory `/opt/EmbedSky/u-boot-1.1.6/post'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/opt/EmbedSky/u-boot-1.1.6/post'
```

Compiling is complete:

```
root@EmbedSky:/opt/EmbedSky/u-boot-1.1.6
文件(F)  编辑(E)  查看(V)  终端(T)  转到(G)  帮助(H)
r - xyzModem.o
r - cmd_mac.o
make[1]: Leaving directory `/opt/EmbedSky/u-boot-1.1.6/common'
UNDEF_SYM=`/opt/EmbedSky/crosstools_3.4.1_softfloat/arm-linux/gcc-3.4.1-glibc-2.
3.3/bin/arm-linux-objdump -x lib_generic/libgeneric.a board/SKY2440/libSKY2440.a
 cpu/arm920t/libarm920t.a cpu/arm920t/s3c24x0/libs3c24x0.a lib_arm/libarm.a fs/c
ramfs/libcramfs.a fs/fat/libfat.a fs/fdos/libfdos.a fs/jffs2/libjffs2.a fs/reise
rfs/libreiserfs.a fs/ext2/libext2fs.a net/libnet.a disk/libdisk.a rtc/librtc.a d
tt/libdtt.a drivers/libdrivers.a drivers/nand/libnand.a drivers/nand_legacy/libn
and_legacy.a drivers/usb/libusb.a wince/libwince.a common/libcommon.a |sed  -n -
e 's/.*\(__u_boot_cmd_.*\)/-u\1/p'|sort|uniq`;\
        cd /opt/EmbedSky/u-boot-1.1.6 && /opt/EmbedSky/crosstools_3.4.1_softfloa
t/arm-linux/gcc-3.4.1-glibc-2.3.3/bin/arm-linux-ld -Bstatic -T /opt/EmbedSky/u-b
oot-1.1.6/board/SKY2440/u-boot.lds -Ttext 0x33F80000  $UNDEF_SYM cpu/arm920t/sta
rt.o \
                --start-group lib_generic/libgeneric.a board/SKY2440/libSKY2440.
a cpu/arm920t/libarm920t.a cpu/arm920t/s3c24x0/libs3c24x0.a lib_arm/libarm.a fs/
cramfs/libcramfs.a fs/fat/libfat.a fs/fdos/libfdos.a fs/jffs2/libjffs2.a fs/reis
erfs/libreiserfs.a fs/ext2/libext2fs.a net/libnet.a disk/libdisk.a rtc/librtc.a
dtt/libdtt.a drivers/libdrivers.a drivers/nand/libnand.a drivers/nand_legacy/lib
nand_legacy.a drivers/usb/libusb.a wince/libwince.a common/libcommon.a --end-gro
up -L /opt/EmbedSky/crosstools_3.4.1_softfloat/arm-linux/gcc-3.4.1-glibc-2.3.3/l
ib/gcc/arm-linux/3.4.1 -lgcc \
                -Map u-boot.map -o u-boot
/opt/EmbedSky/crosstools_3.4.1_softfloat/arm-linux/gcc-3.4.1-glibc-2.3.3/bin/arm
-linux-objcopy --gap-fill=0xff -O srec u-boot u-boot.srec
/opt/EmbedSky/crosstools_3.4.1_softfloat/arm-linux/gcc-3.4.1-glibc-2.3.3/bin/arm
-linux-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
[root@EmbedSky u-boot-1.1.6]#
```

Caution: Make sure that the cross-compiler supports softfloat, otherwise the compiling would probably fail.

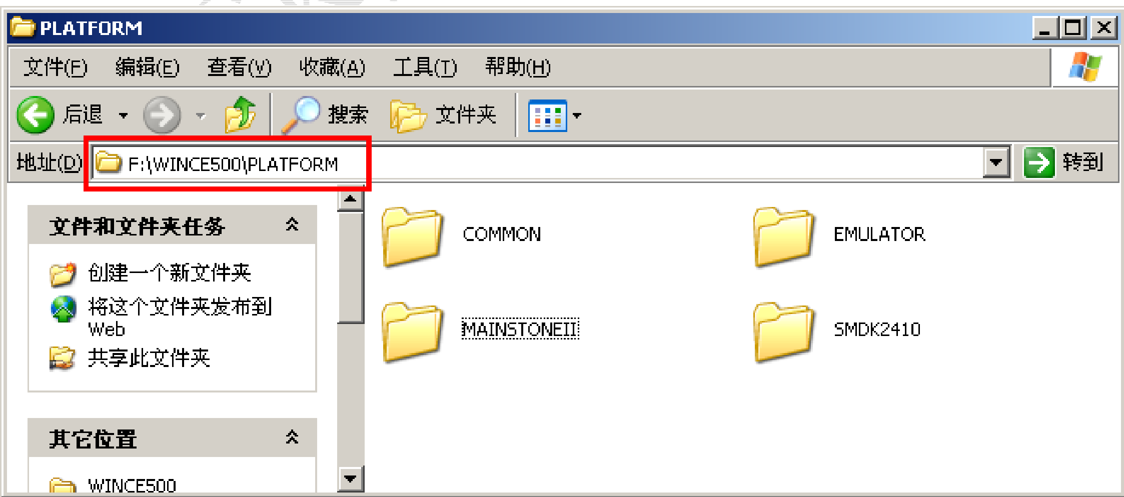
Chapter 5 WinCE development manual

5. 1 Installing BSP of TQ2440

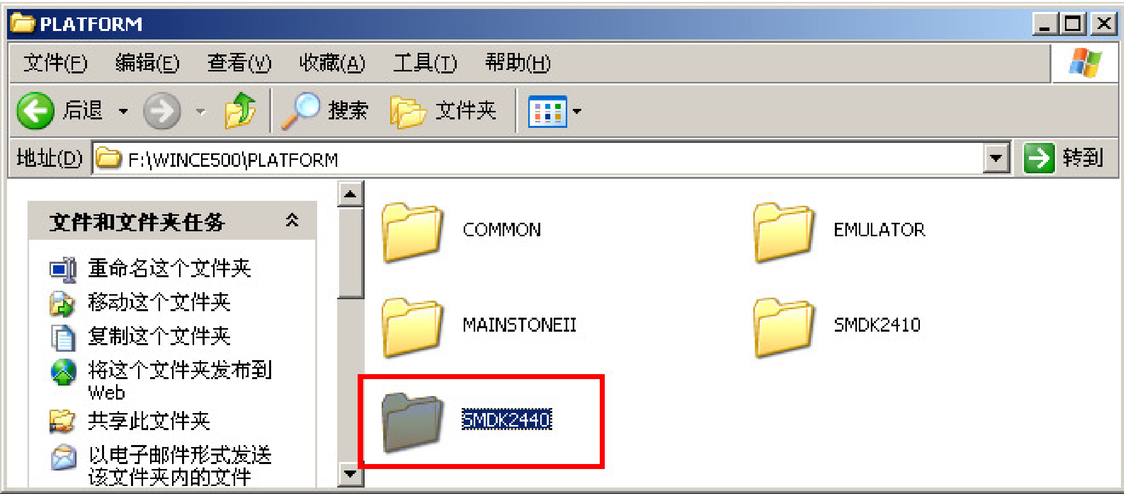
TQ2440 BSP needs to be configured when using WinCE image compiled by PB.

The following contents introduce the installation steps:

Step 1, open the directory “F:\WINCE500\PLATFORM”:

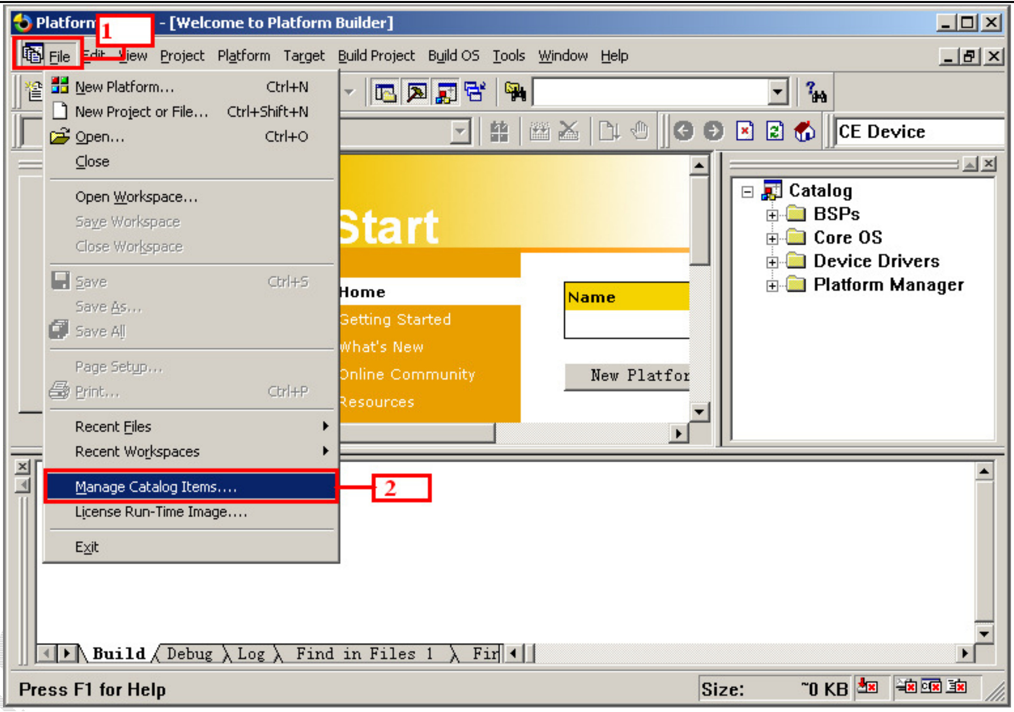


Step 2, copy “WinCE\WinCE_5.0\SMDK2440” in CD-ROM to the directory “WINCE500\PLATFORM”:

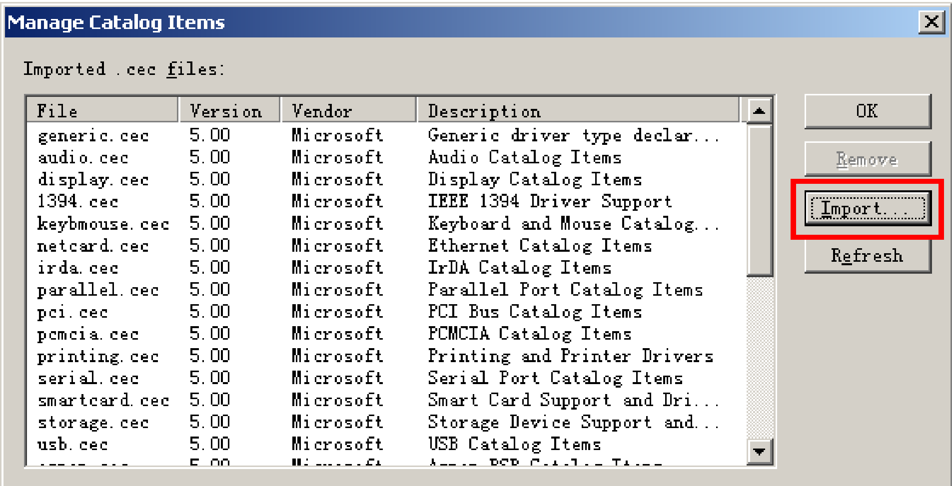


Step 3, remove the read-only property of all files under the directory “SMDK2440”;

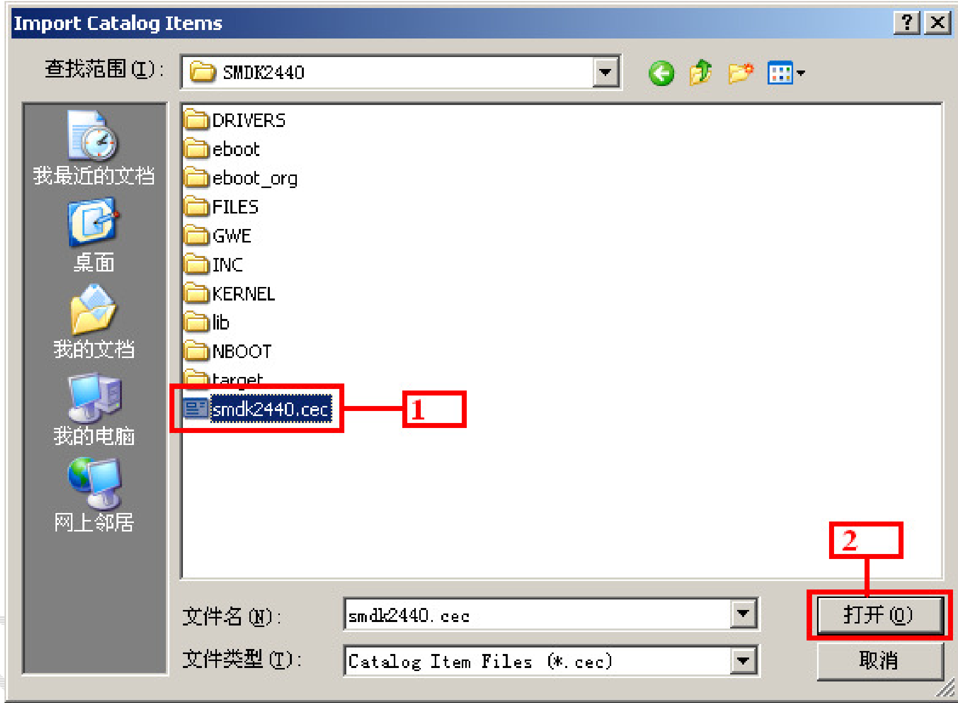
Step 4, run “Platform Builder 5.0”, select “Manage Catalog Features” of the menu “File”, and get into BSP package manager menu:



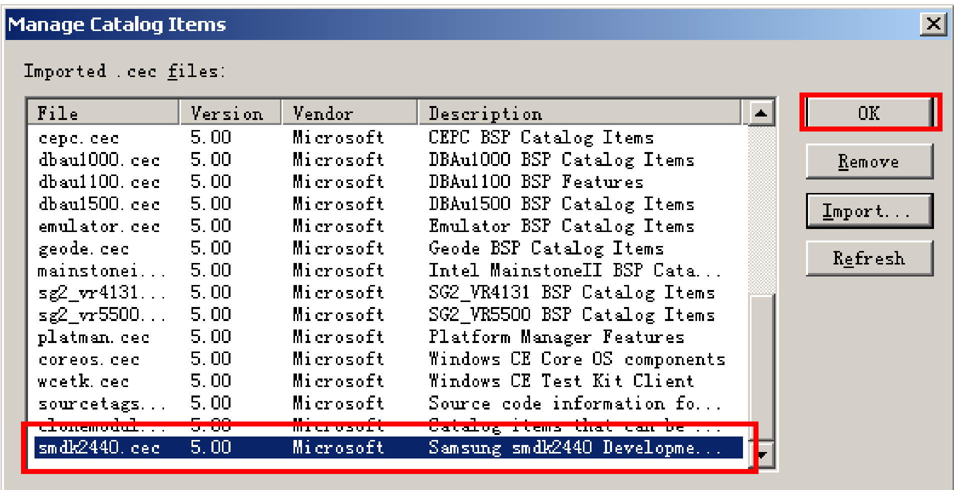
Step 5, select “Import” to load the file “F:\WINCE500\PLATFORM\SMDK2440\smdk2440.cec” in the following Manage Catalog Features interface:



Click “import” and open “smdk2440.cec”:

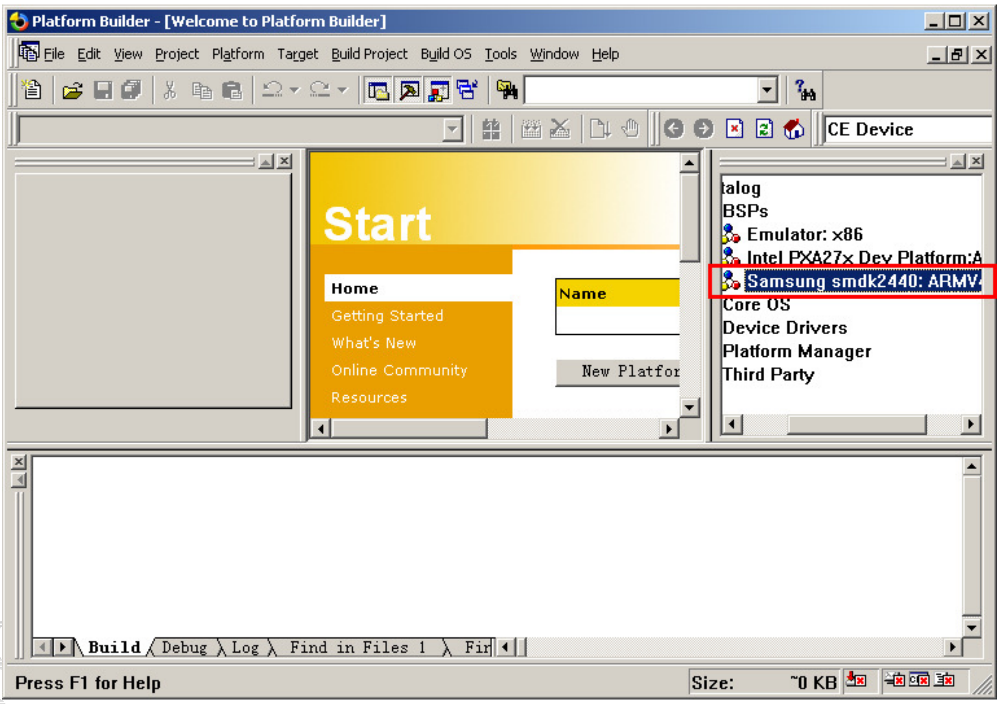


Select “smdk2440.cec” and click “OK” to continue.



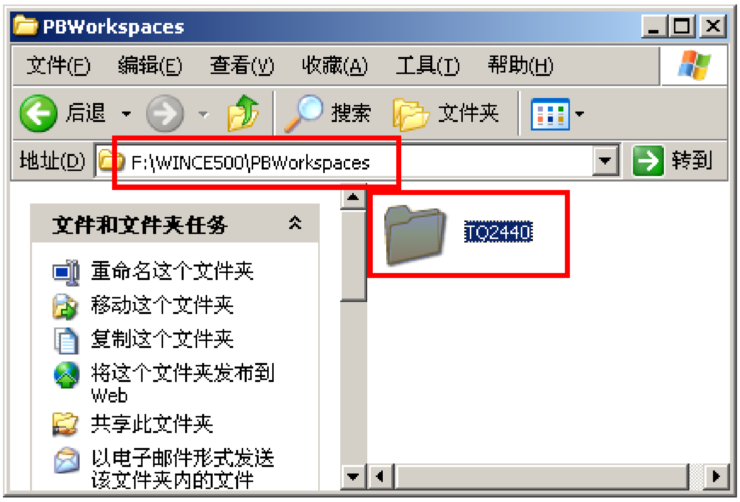
Step 6, after BSP installation, the option “Samsung SMDK2440：ARMV4I” is auto-added to the PB interface, below the option “BSPs” of “Catalog”.

As the following red frame indicates:

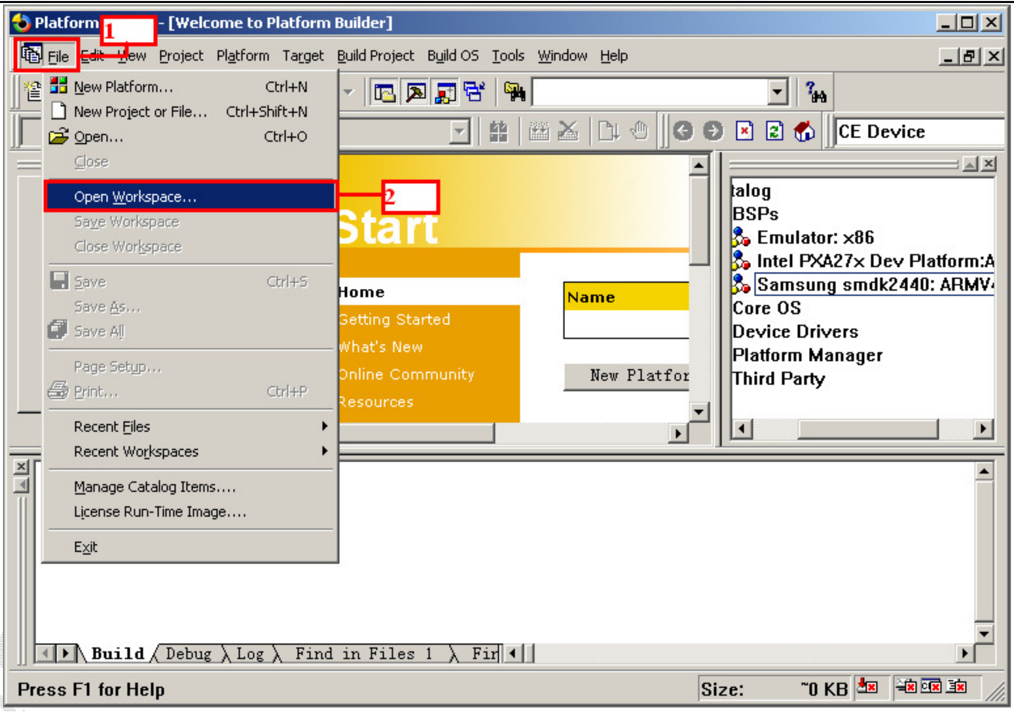


5.2 Compiling the example projects in CD-ROM

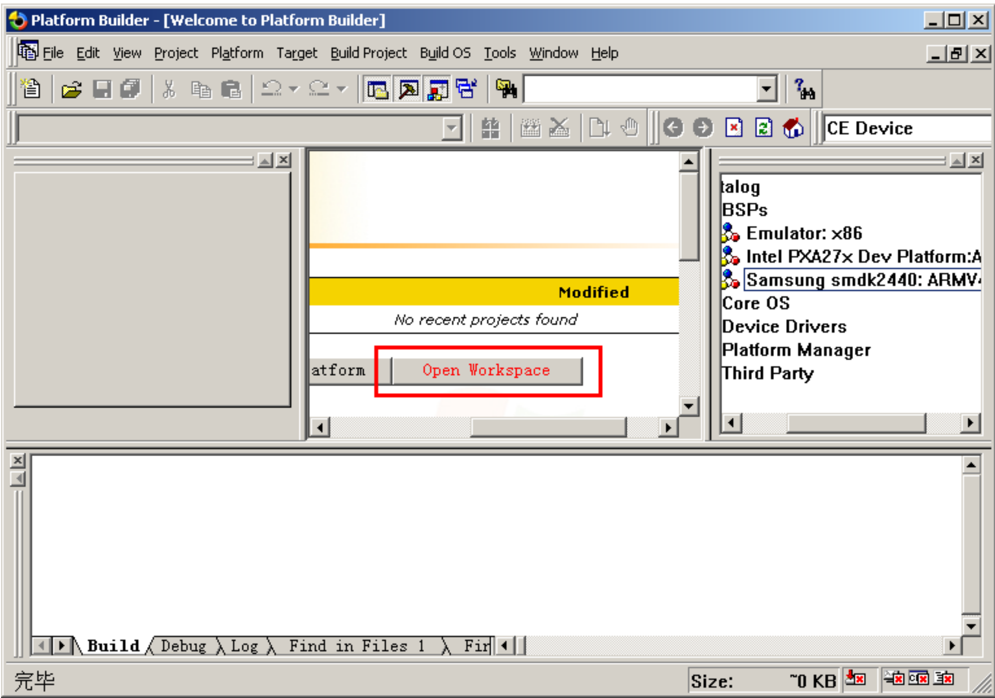
Step1, find the "TQ2440" project files under the directory "WinCE\WinCE_5.0" in CD-ROM. Copy these files to the directory "F:\WINCE500\PBWorkspaces" and remove the read-only property (if there is no PBWorkspaces directory, you can create one.).



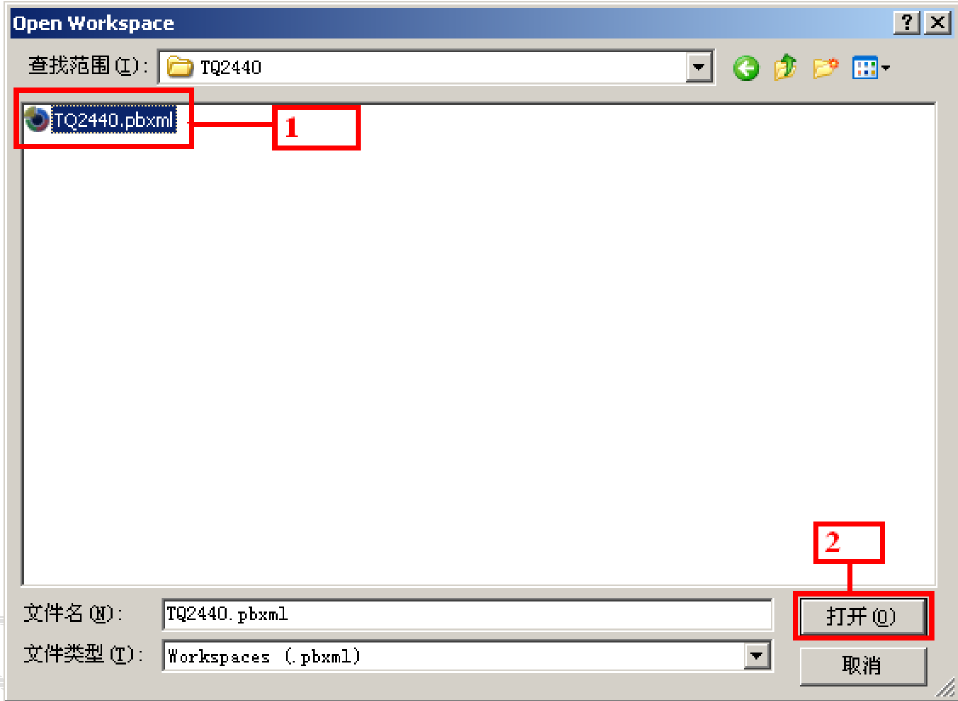
Step2, click "Open Workspace" in the menu "File" or in interface of PB, as the following red frame indicates:



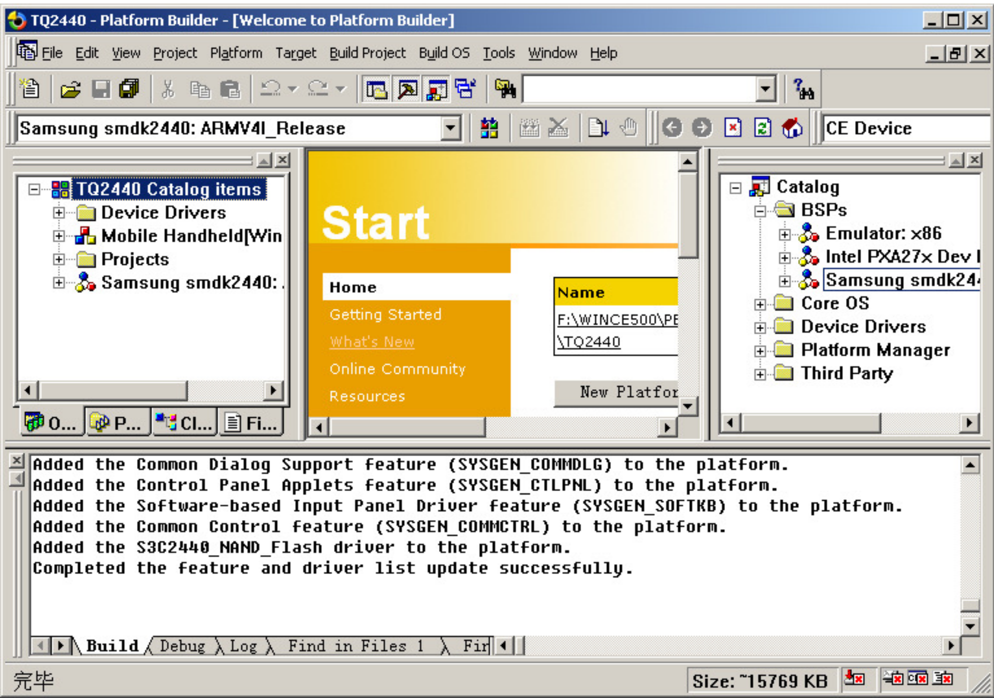
Or:



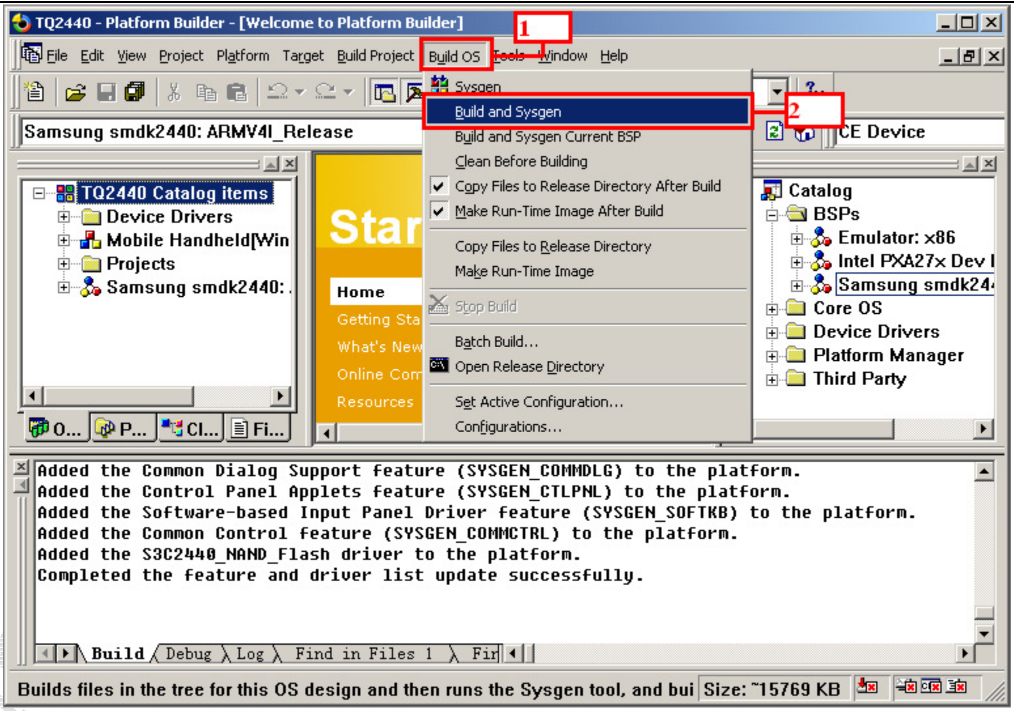
Open the project file “TQ2440” and find “TQ2440.pbxml”:



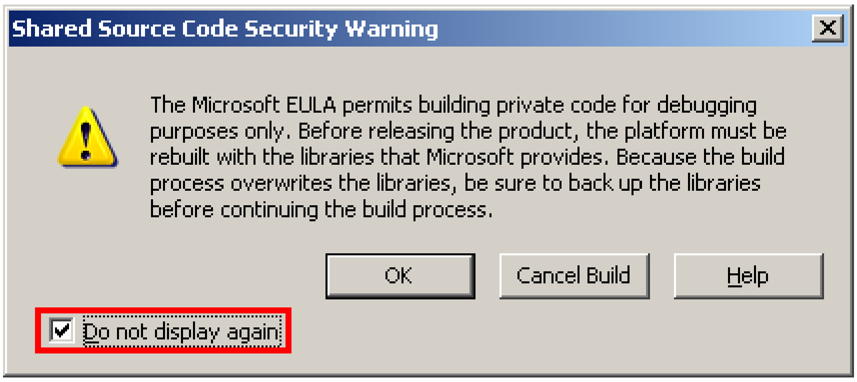
The project:



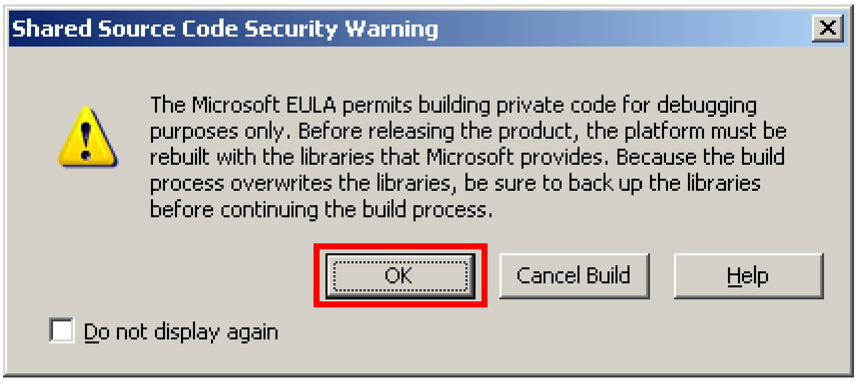
Step3, click “Build and Sysgen” in the menu “Build” to start compiling:



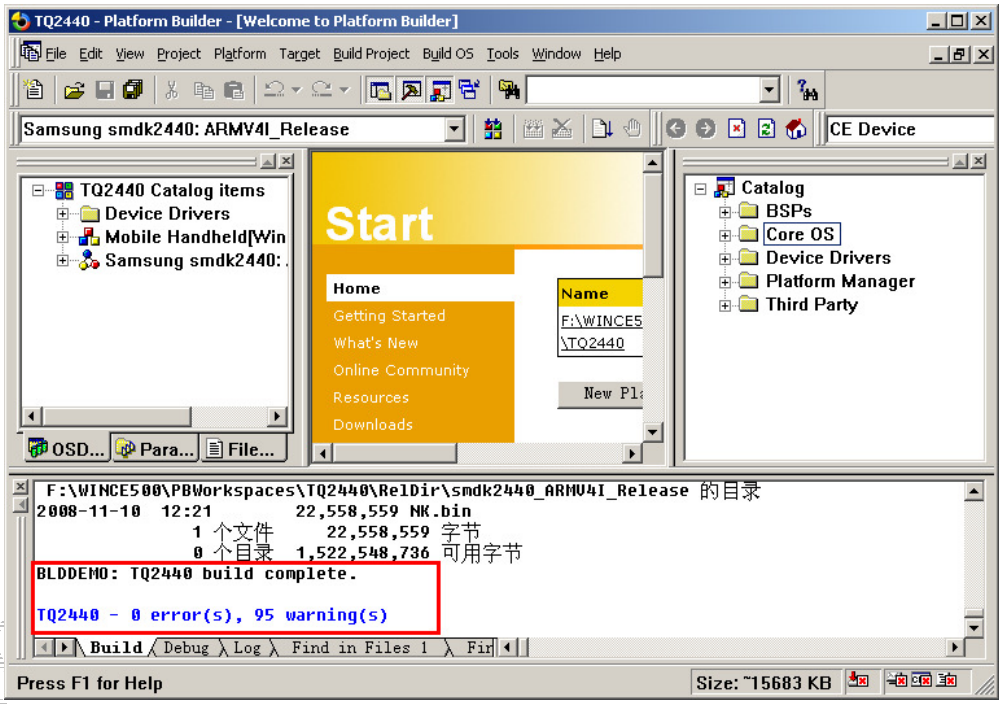
The following interface probably appears, select “Do not display again”:



Click “OK”:



Compiling is complete:

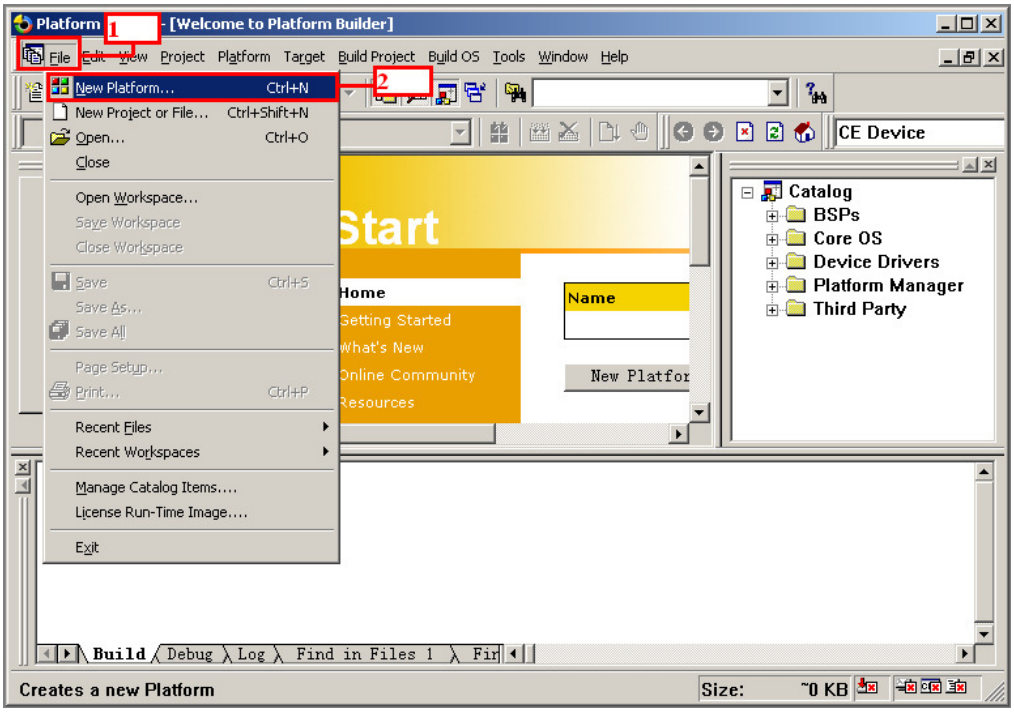


Step4, 2 kernel files “NK.bin” and “NK.nb0” are created after compiling. Only “NK.bin” is useful which is under the directory “F:\WINCE500\PBWorkspaces\TQ2440\RelDir\smdk2440_ARMV4I_Release”.

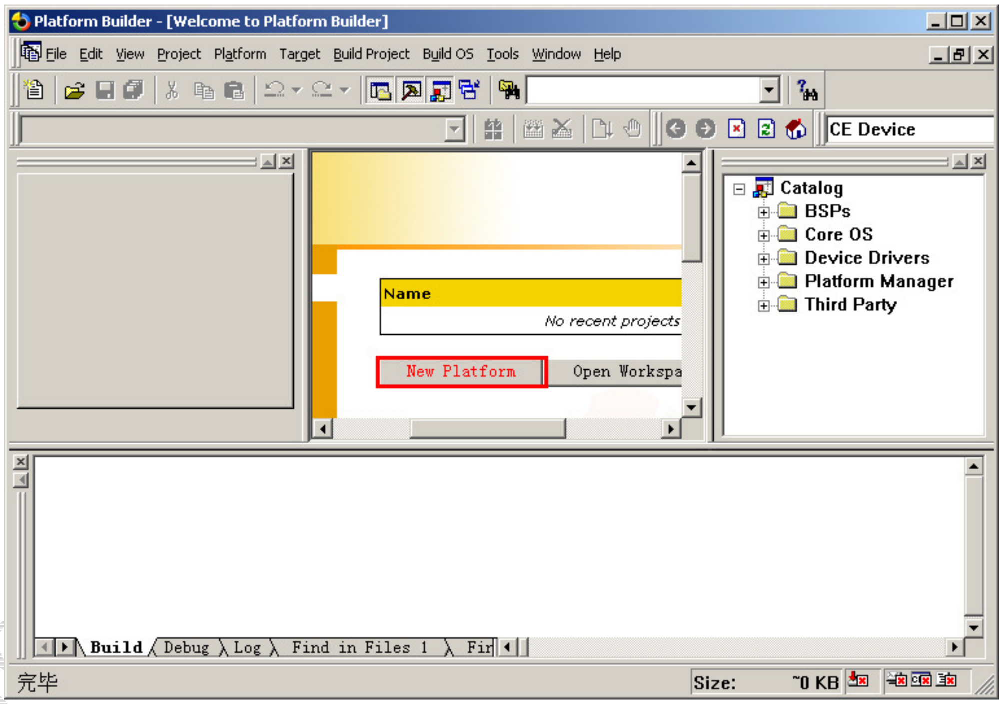
5. 3 Customizing user project files

The following contents introduce how to customize the user project files:

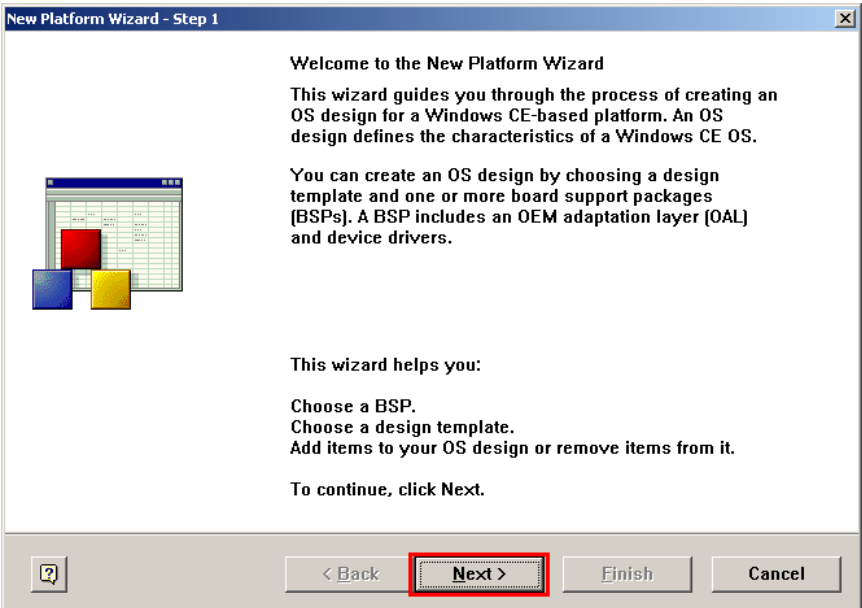
Step 1, click “New Platform” in the menu “File” or in the “Start” interface:



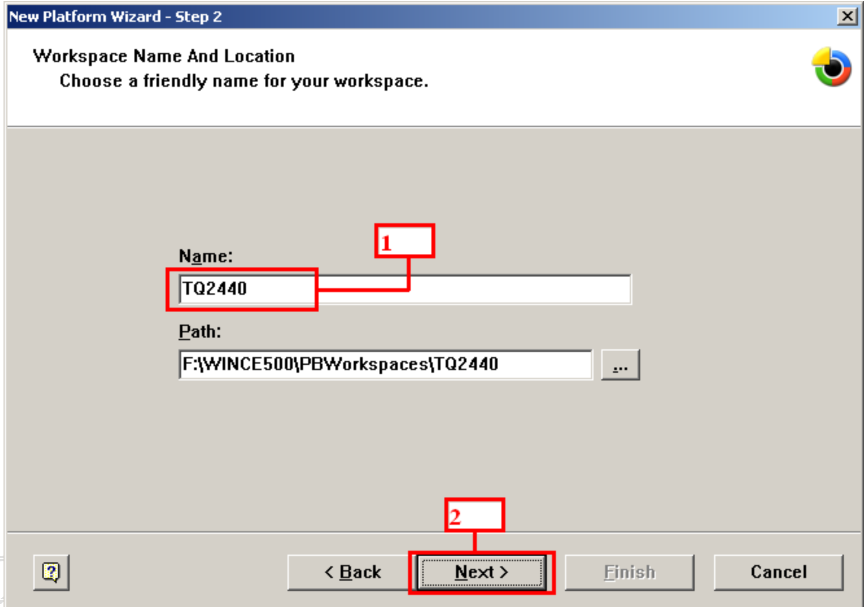
Or:



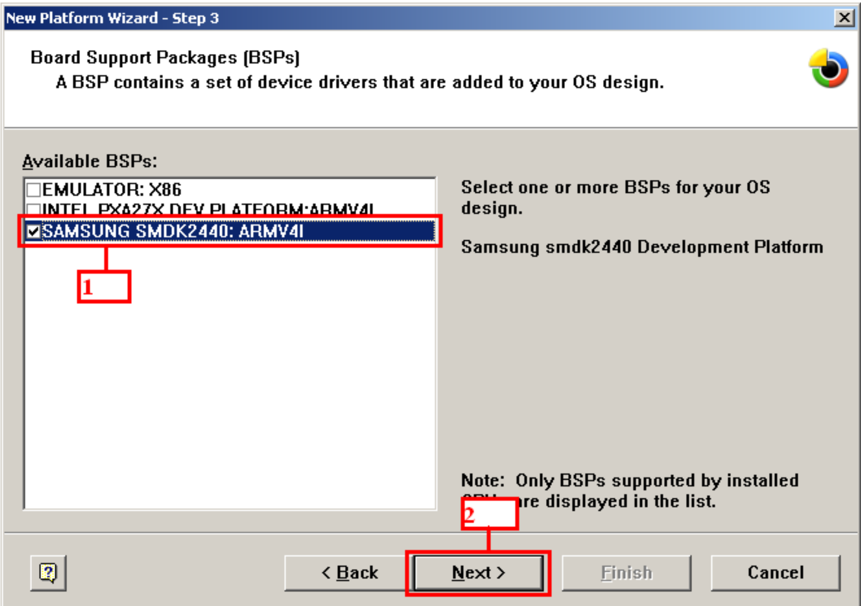
Click “Next” in the following interface“New Platform Wizard”:



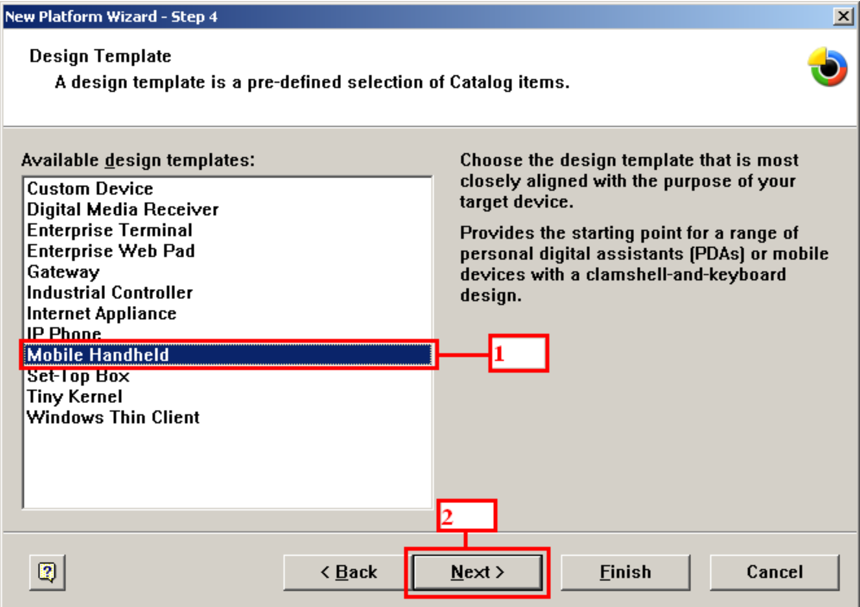
Step 2, input the project name and the location of saved files (usually adopt the default location) in the interface “Workspace Name And Location” and click “Next” to continue:



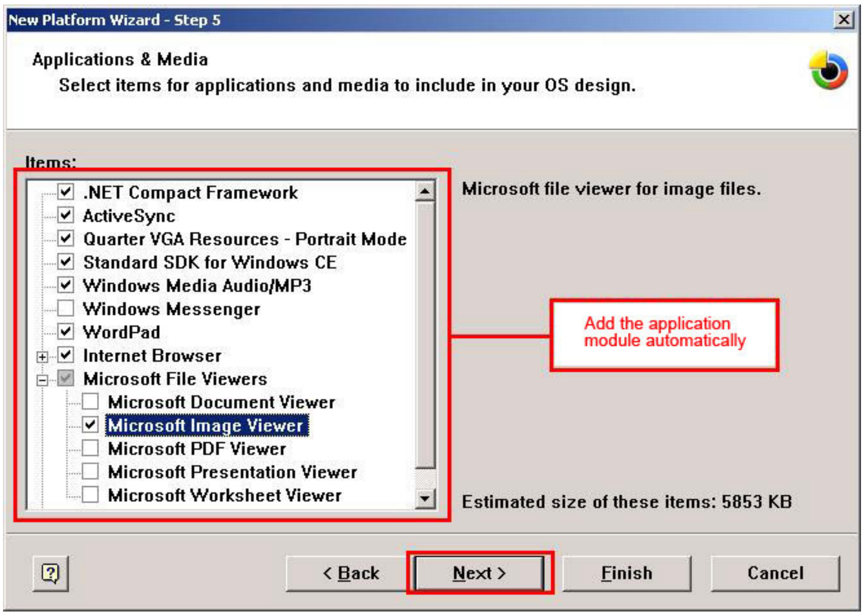
Step 3, select “SAMSUNG SMDK2440：ARMV4I” in the interface “Board Support Package (BSPs)” and click “Next” to continue:



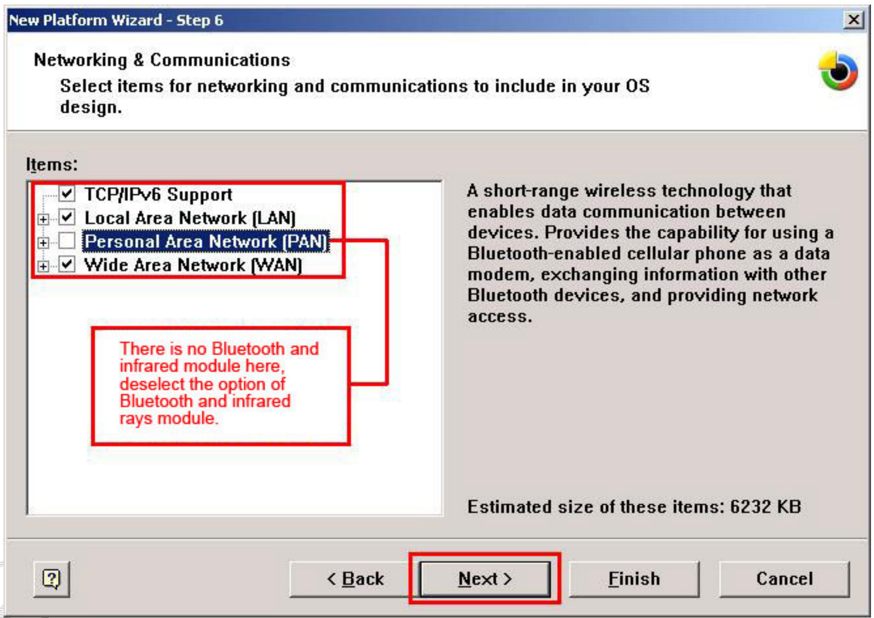
Step 4, select “Mobile Handheld” in the interface “Design Template” and click “Next” to continue:



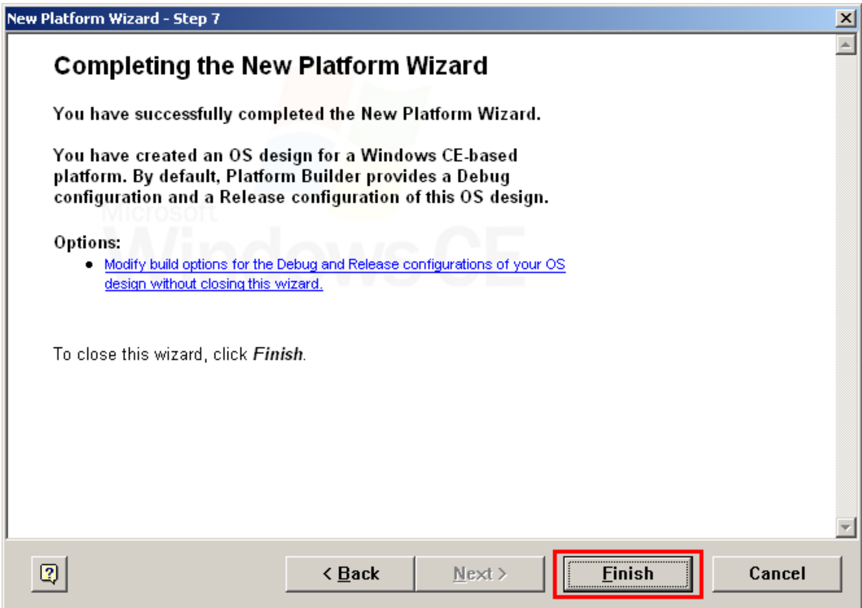
Step 5, select the following options in the interface “Applications & Media” and click “Next” to continue:

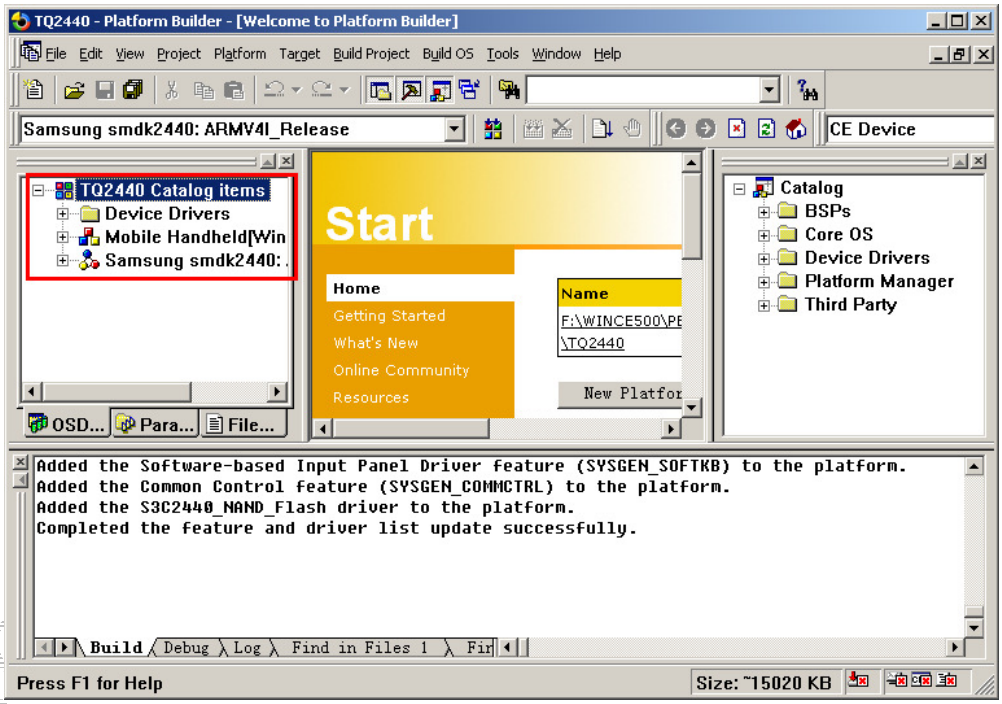


Step 6, adopt the default configurations in the interface “Networking & Communications” and click “Next” to continue:

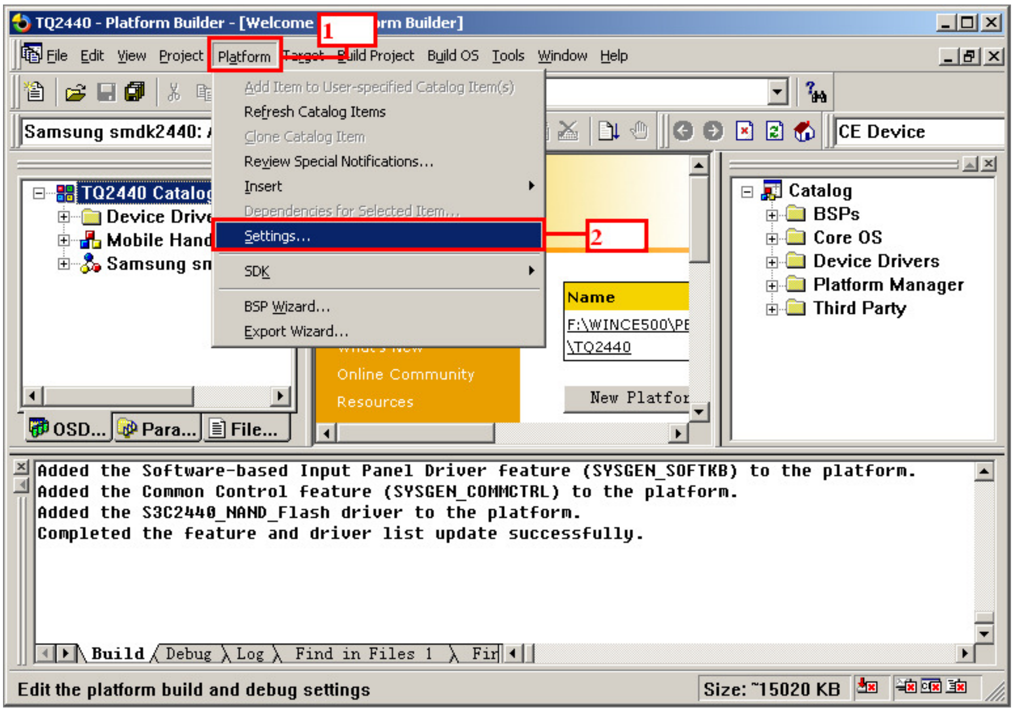


Step 7, click "Finish" in the interface "New Platform Wizard – Step 7" to complete project customizing:

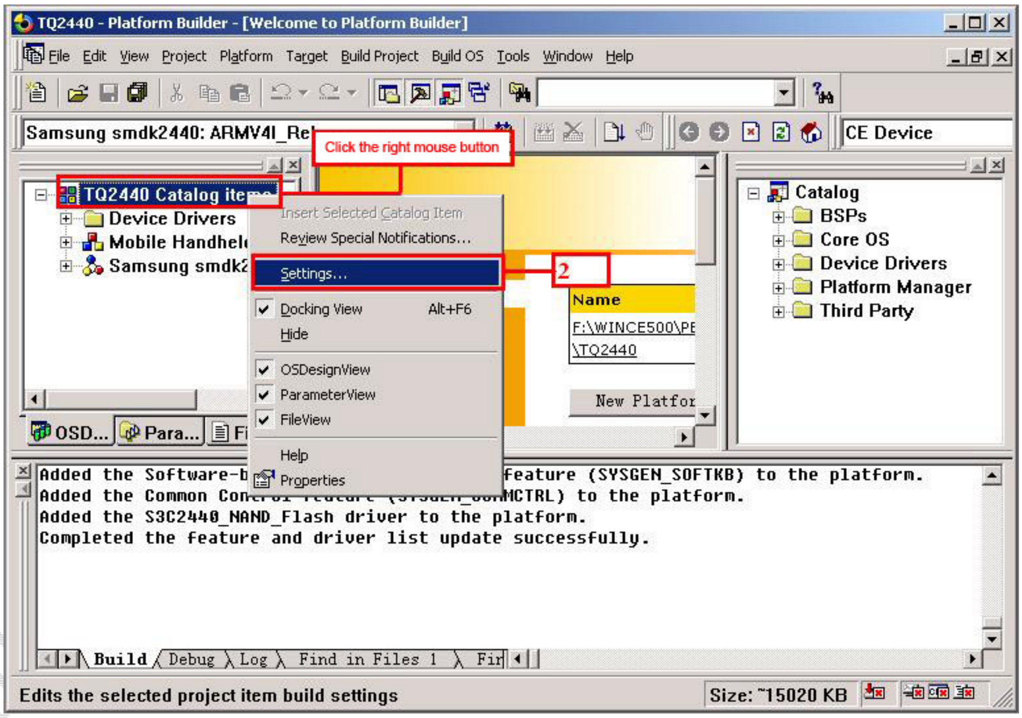




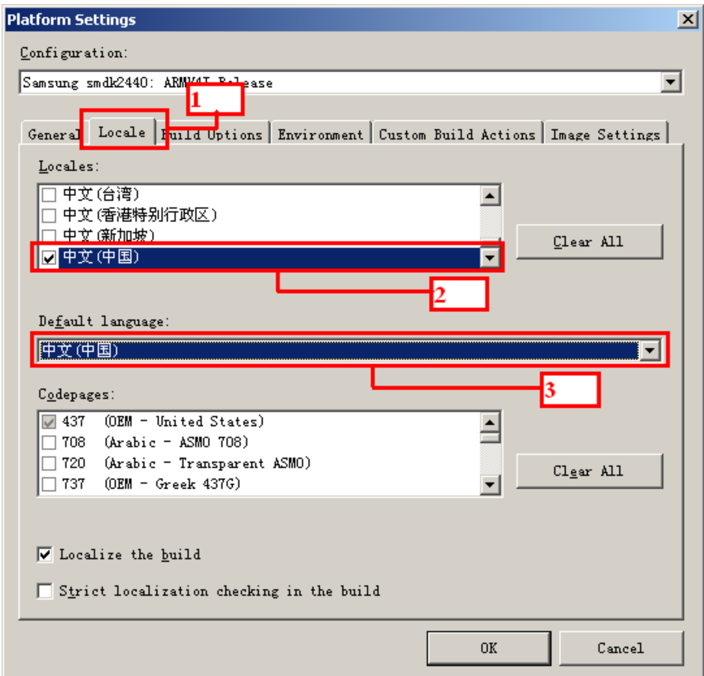
Step 8, click “Settings” in the menu “Platform” or “Settings” after right-click “TQ2440 features”:



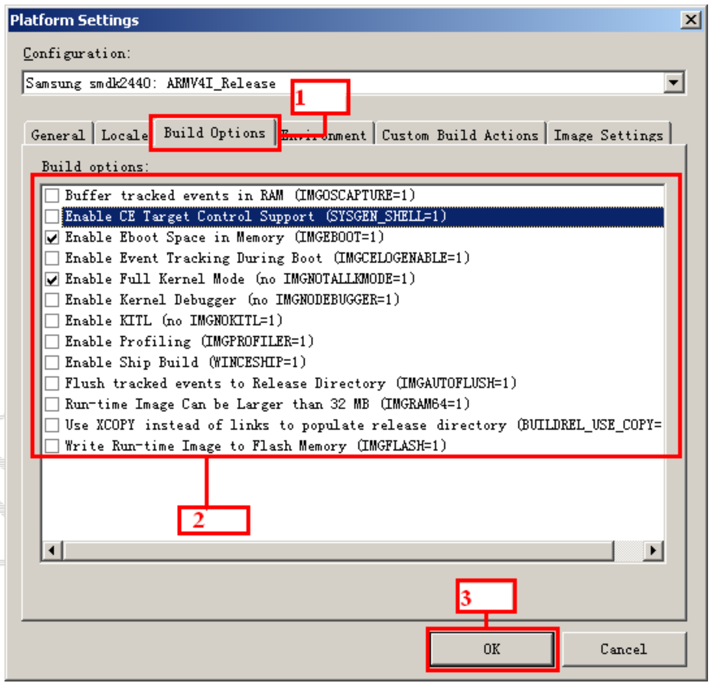
Or:



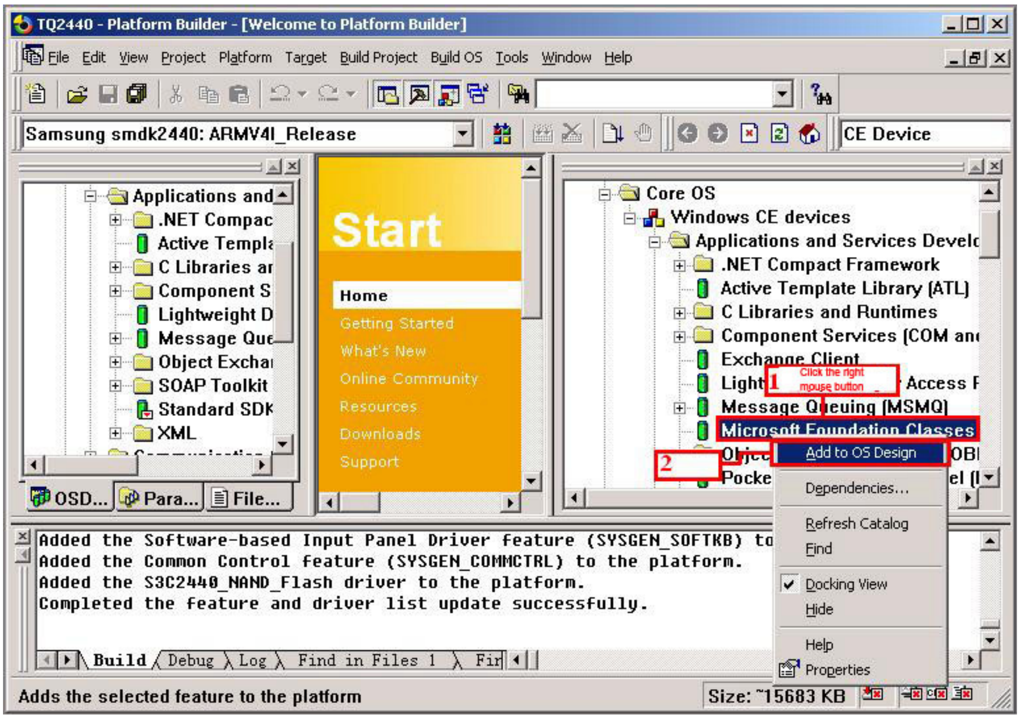
Step 9, select “Locate” page in the interface “Platform Settings” and select Chinese support:



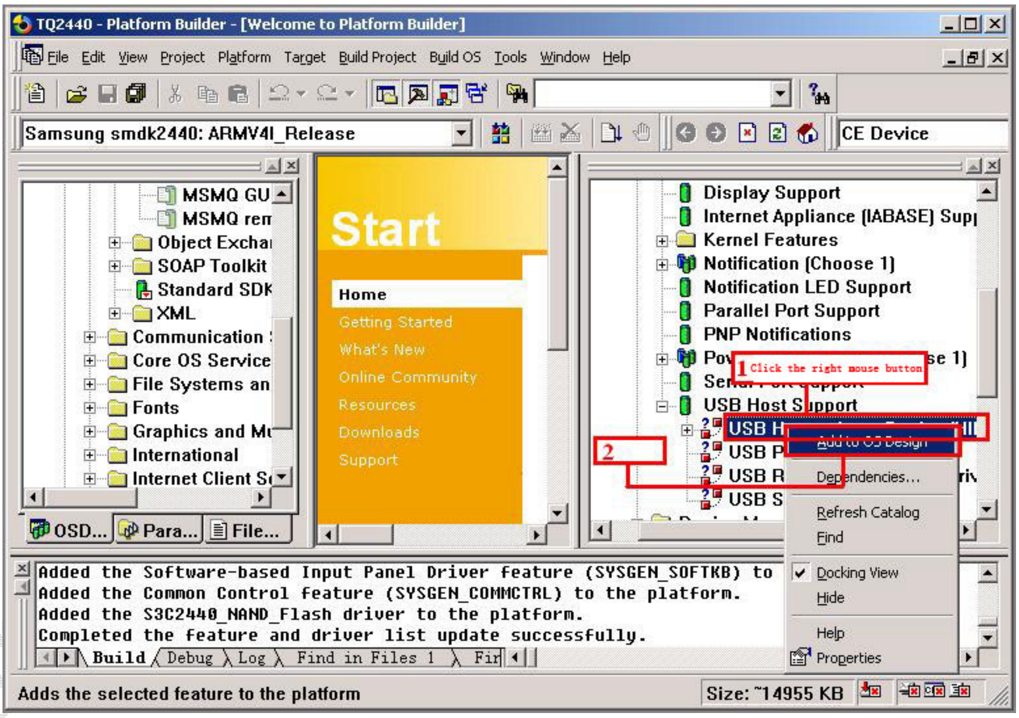
Select the page “Build Options”. De-select “Enable CE Target Control Support” and “Enable KITL” and click “OK” to continue:

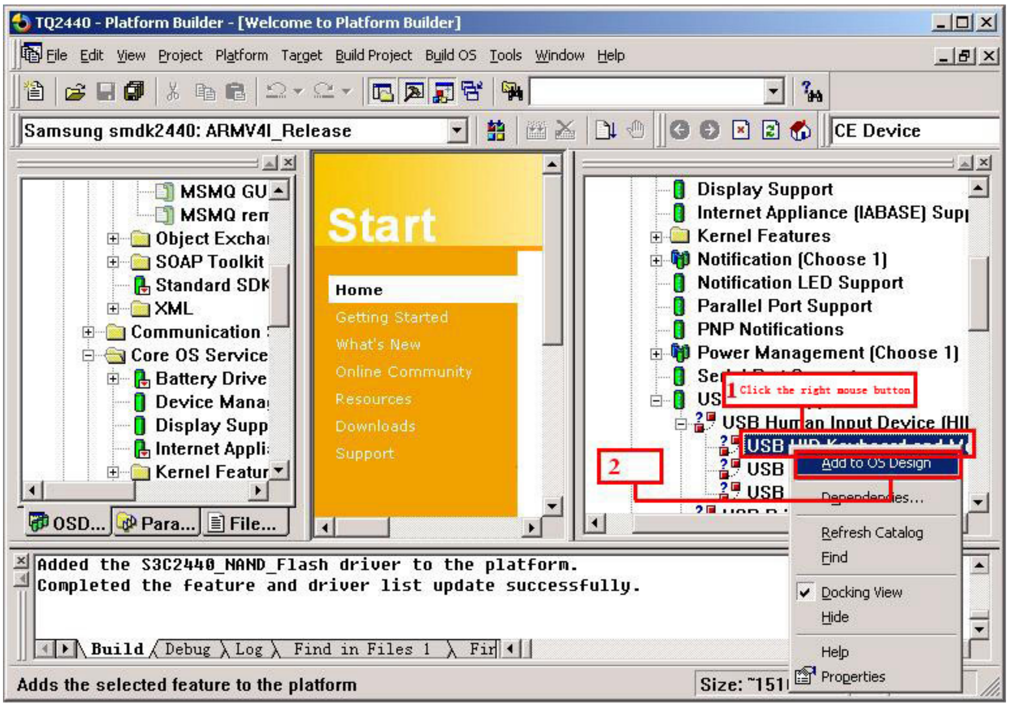


Step10, add MFC components as the following diagram. Open "Catalog->Core OS->Windows CE devices->Applications and Services Devleopment" and right-click the option and select "Add to Platform":

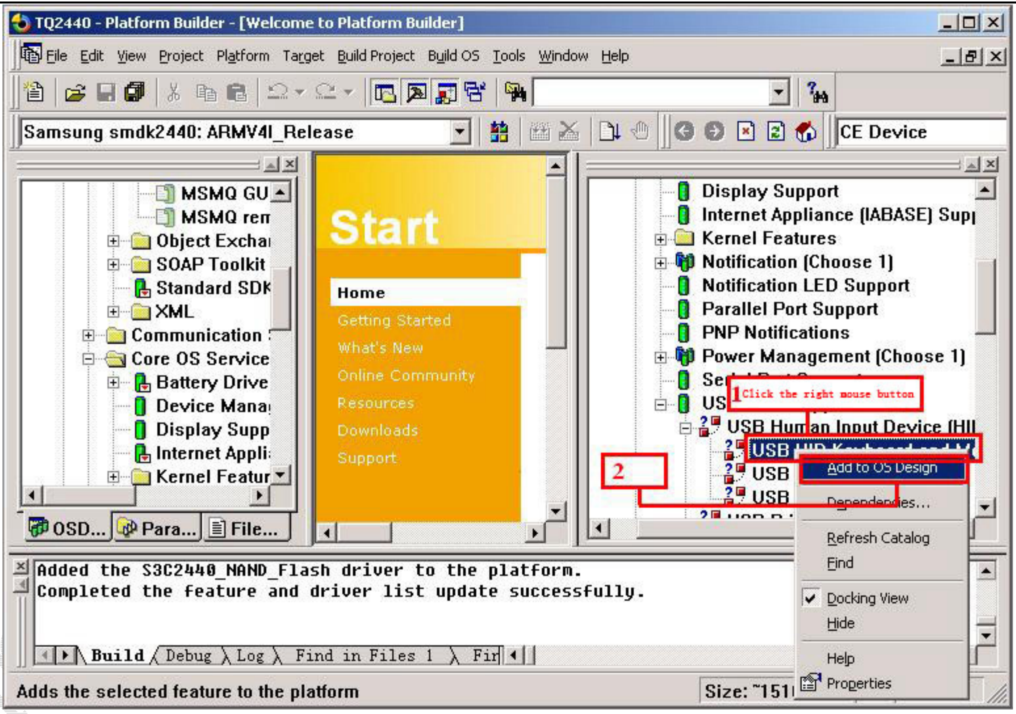


Step 11, add USB keyboard and mouse supporting features. Open "Catalog->Core OS->Windows CE devices->Core OS Service->USB Host Support->USB Human Input Device (HID) Class Driver", right-click it and select "Add to Platform". Right-click "USB HID Keyboard and Mouse" and select "Add to Platform":

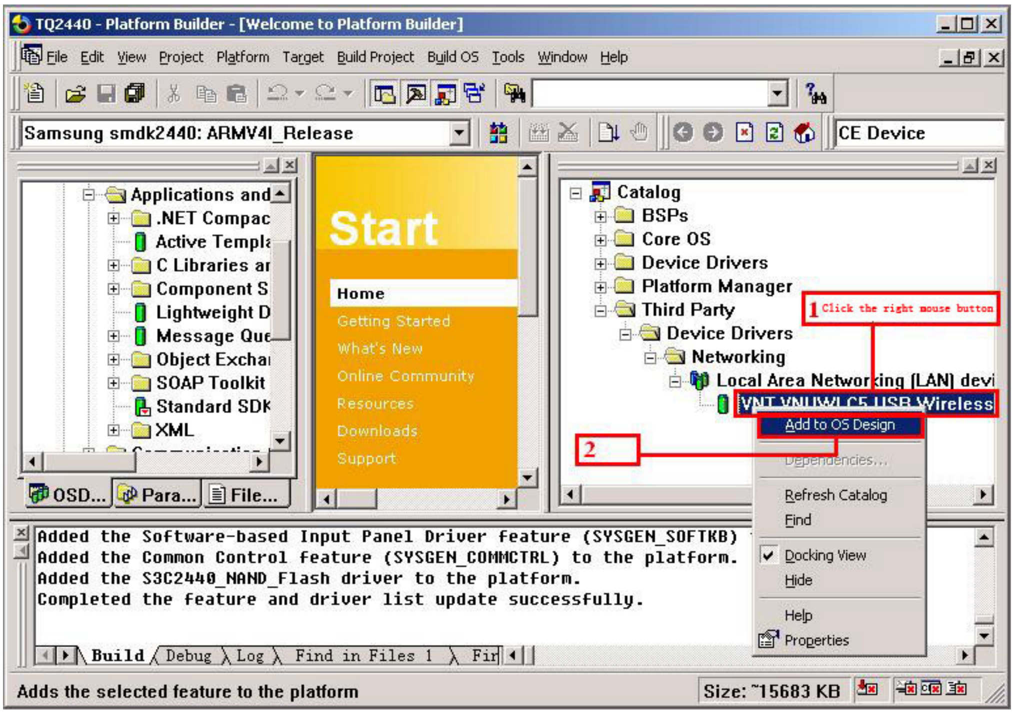




Step 12, add USB mobile storage device supporting features:

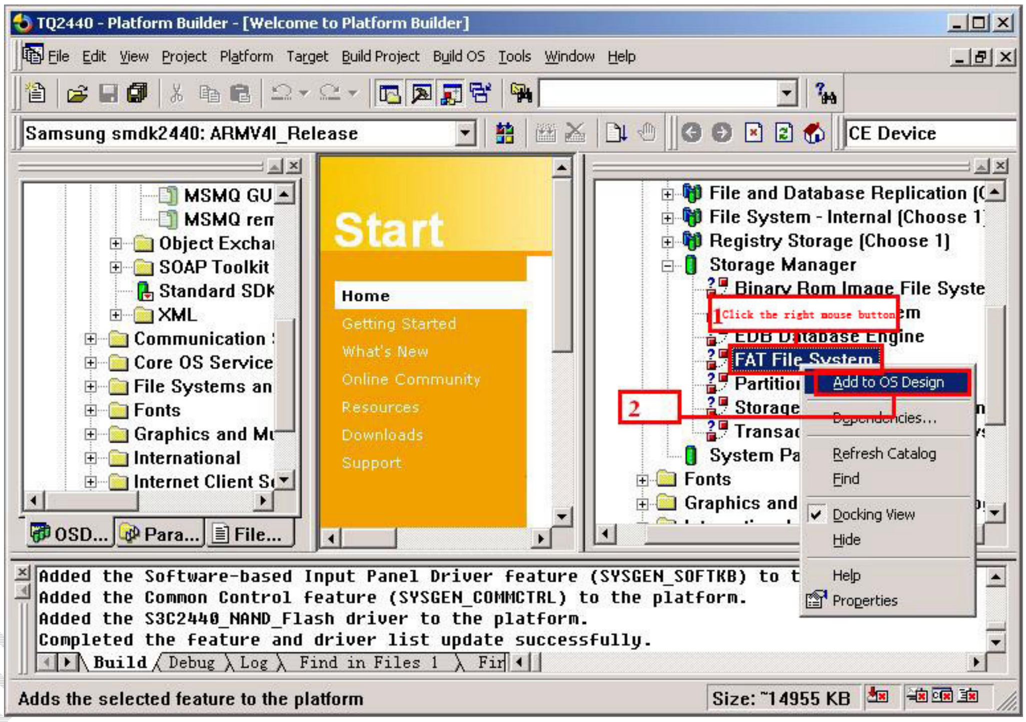


Step 13, add USB interface wireless network device supporting features:

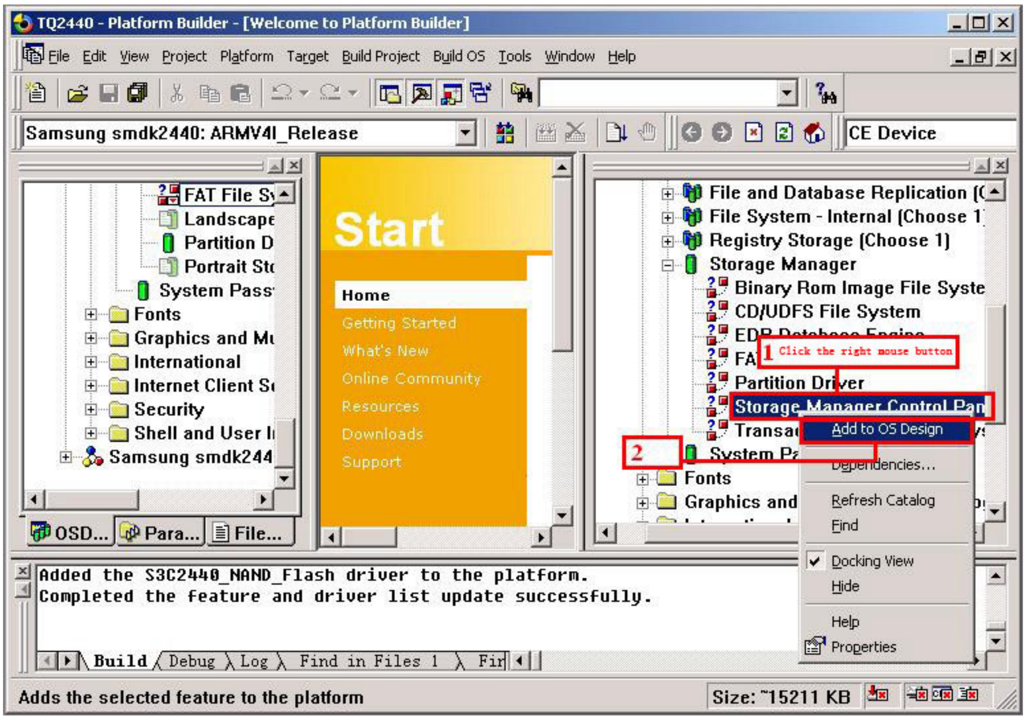


Step14, add file system supporting features. The register can be saved only when HIVE is supported by file system. The following contents introduce the process of file system configuration:

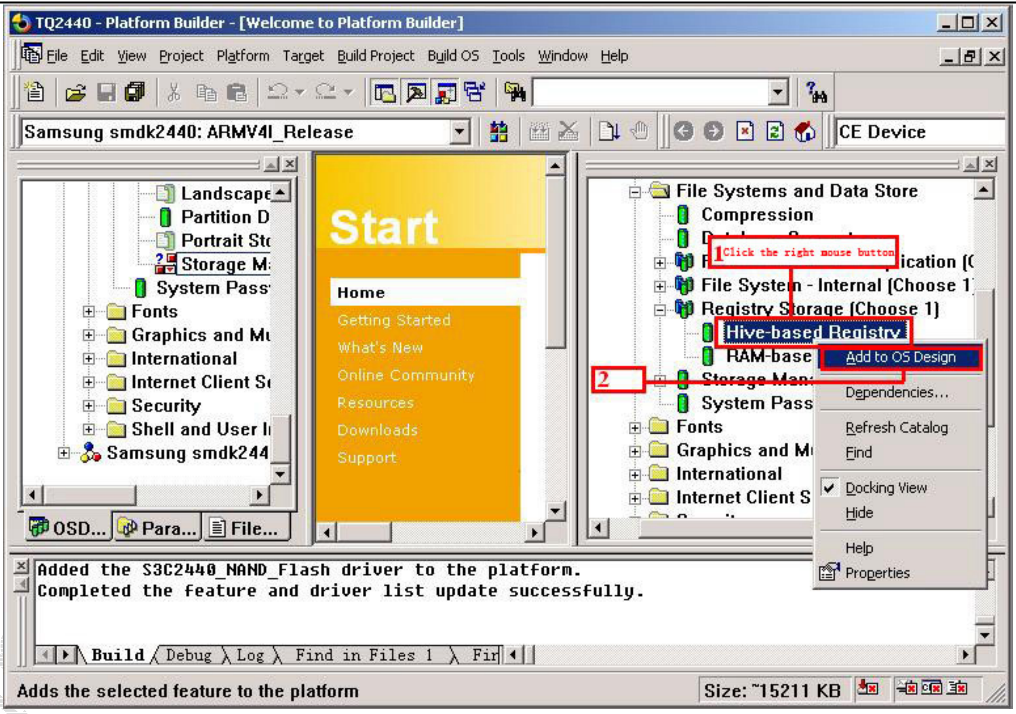
Add FAT file system supporting feature:



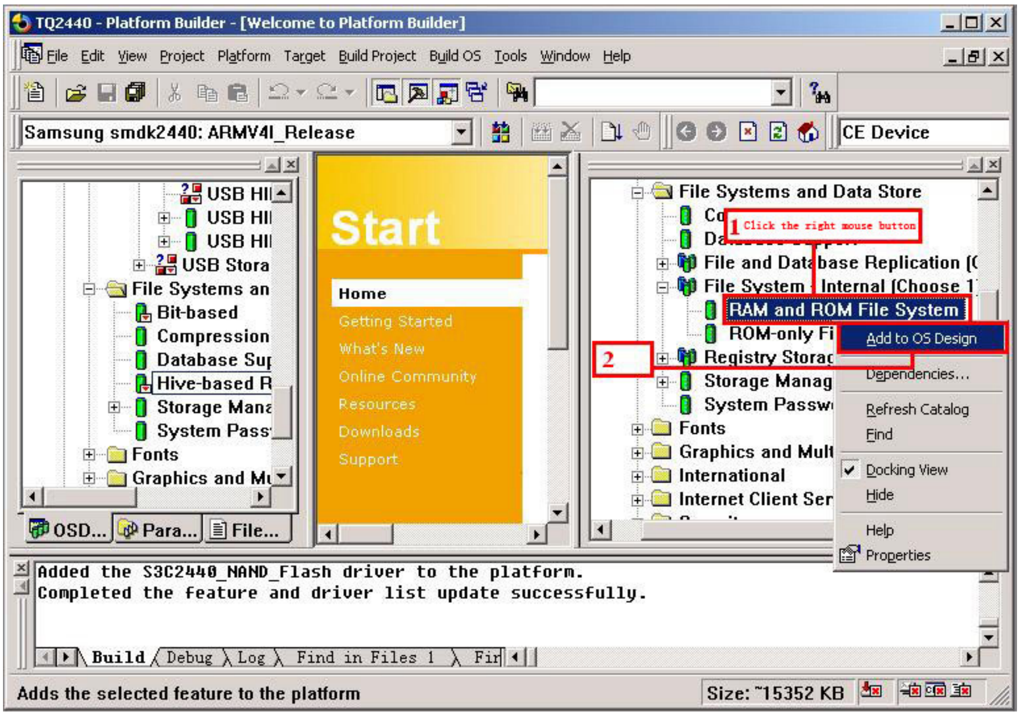
Add “Storage Manager Control Panel Applet”



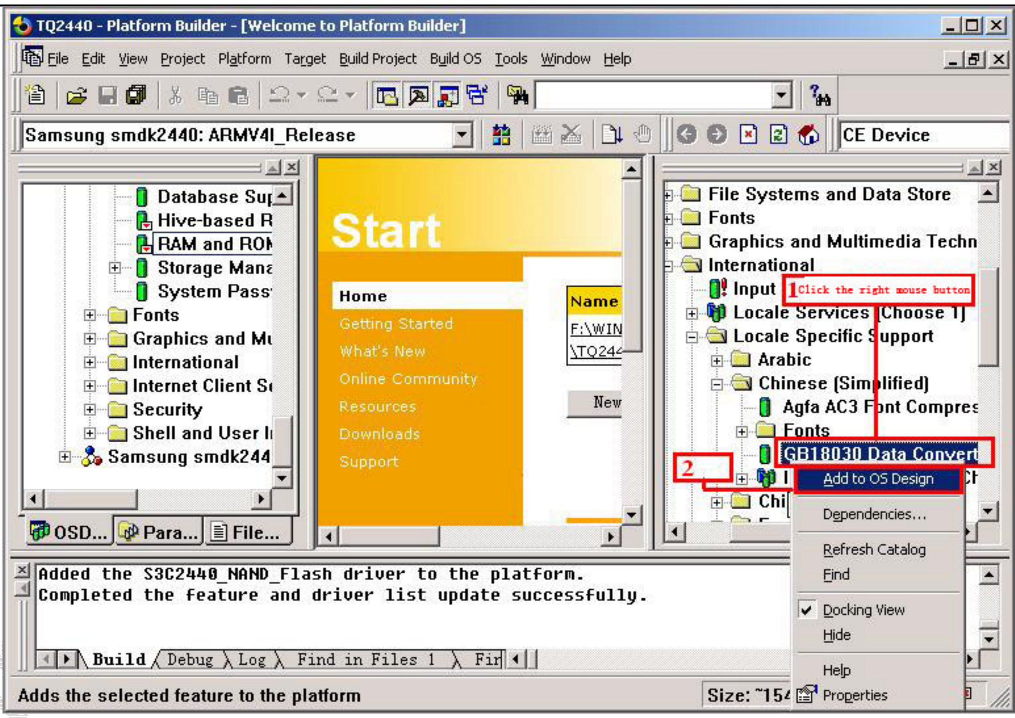
Add HIVE supporting feature:



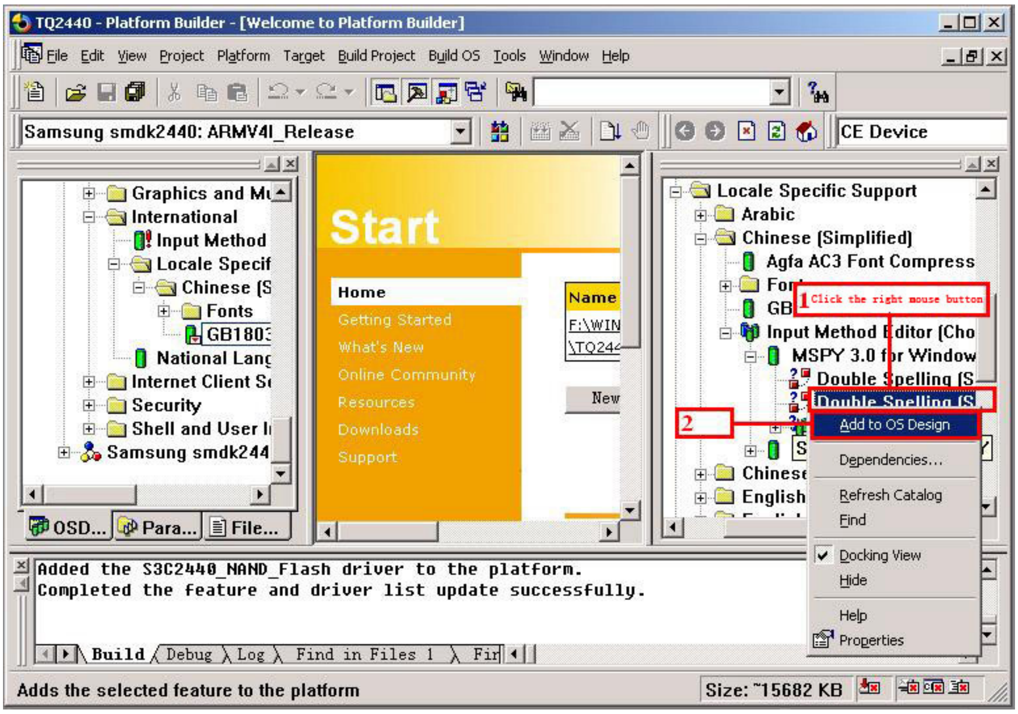
Add RAM and ROM file system supporting feature:



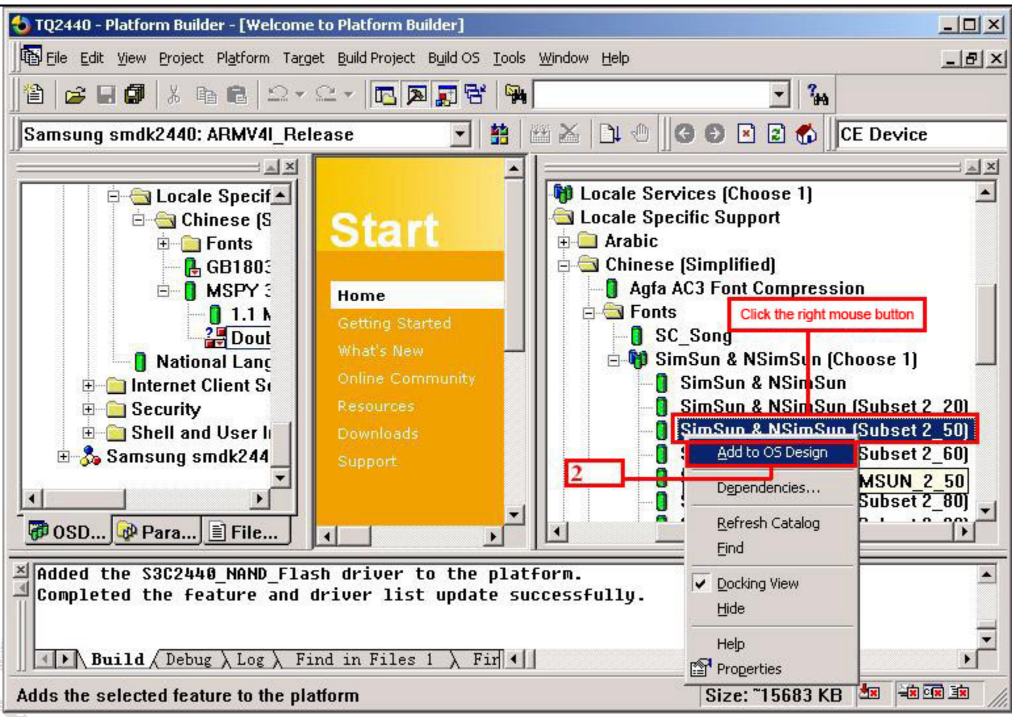
Step 15, add font and input method supporting features:



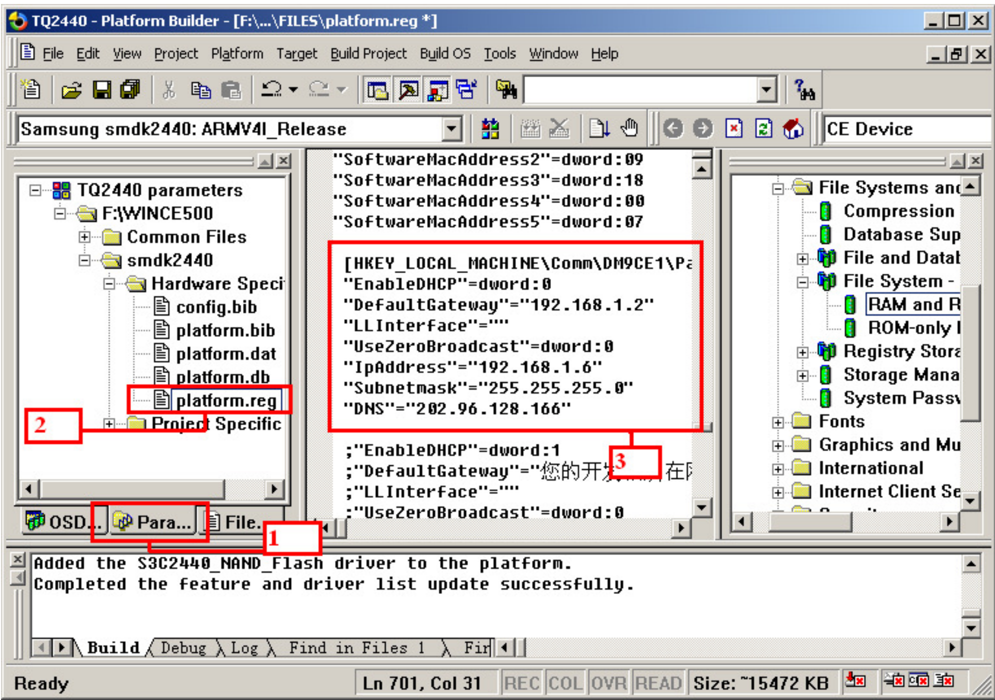
Add PINYIN input method supporting feature:



Add font supporting feature:

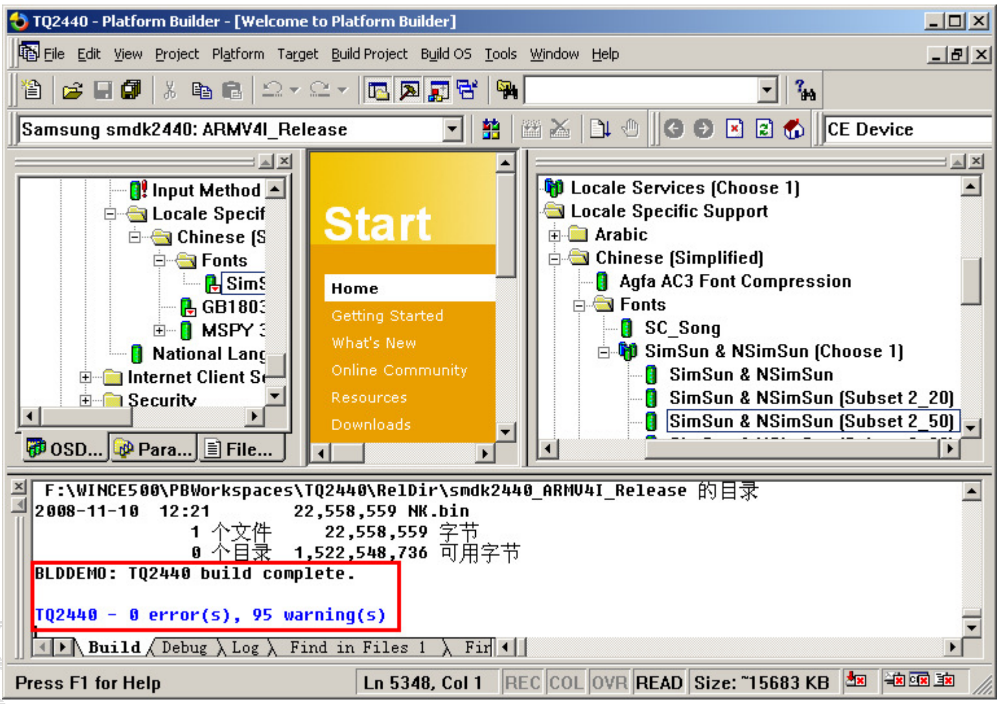


Step 16, configure network parameters: IP address, DNS, gateway and so on by modifying "platform.reg" reference key values.



Step 17, compile the project. Click "Build and Sysgen" in the menu "Build" to start compiling:

Compiling is complete:



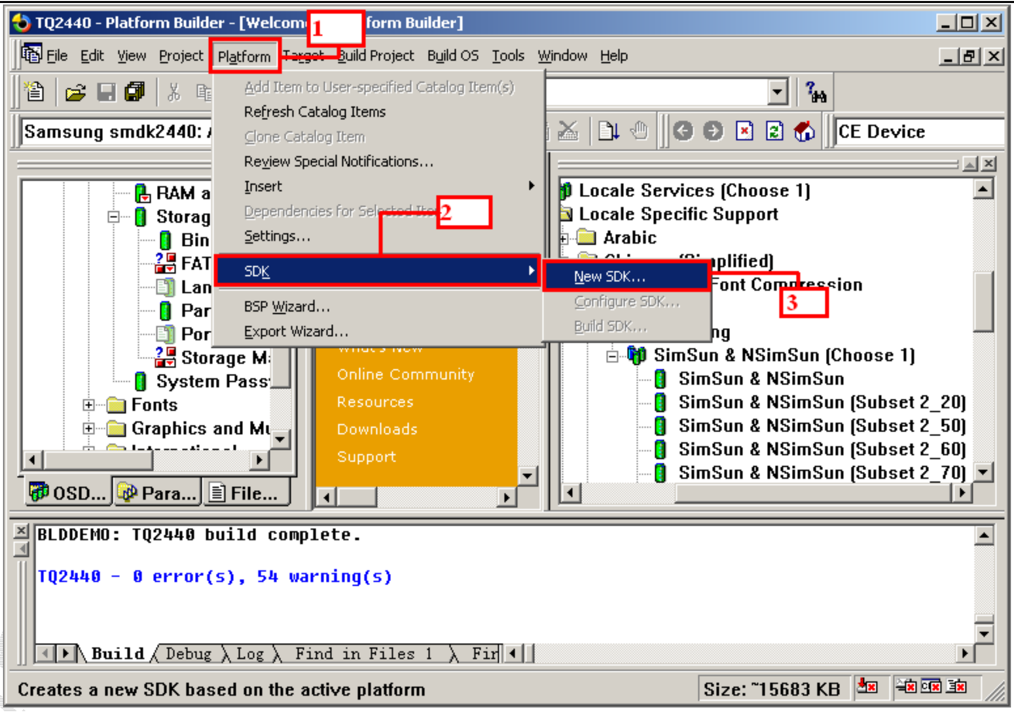
5.4 Export SDK

The users can export the specific SDK for their customized platforms. SDK is the general name of head file, library file, document, platform manager and dynamic link library.

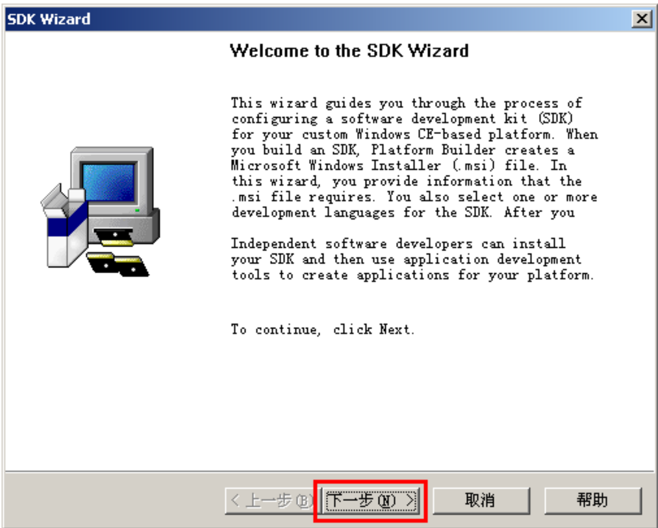
The developer can use SDK to program for a certain platform. This part of chapter introduces the process exporting SDK.

5.4.1 Configure SDK

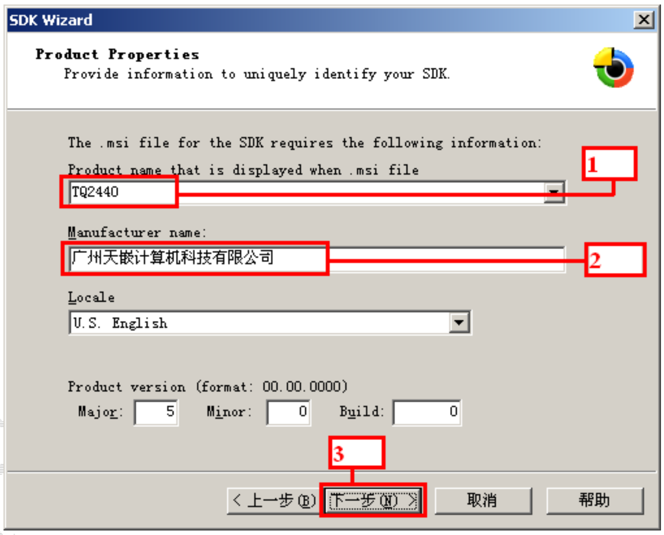
Click "New SDK" of "SDK" in the menu "Platform":



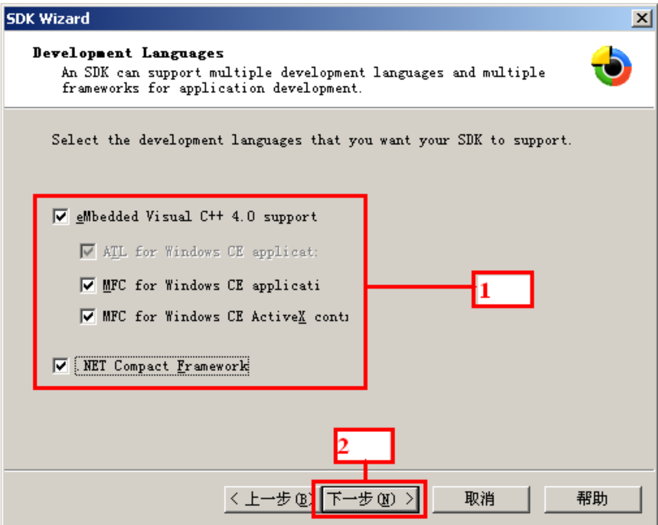
Click “下一步” in the interface “SDK Wizard”:



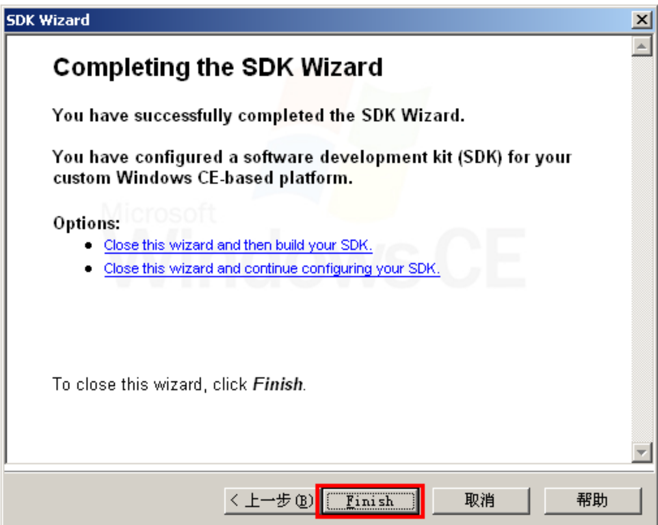
Enter the project name and company in the interface “Product Properties” and click “下一步” to continue:



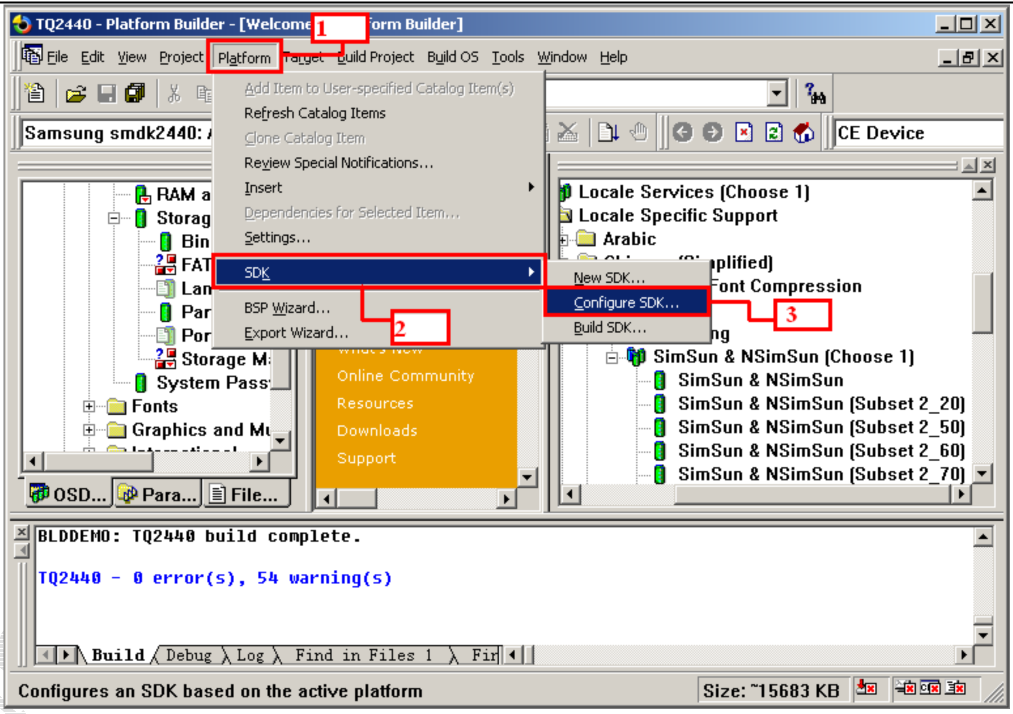
Select the library and development languages in the interface “Development Languages” and click “下一步” to continue:



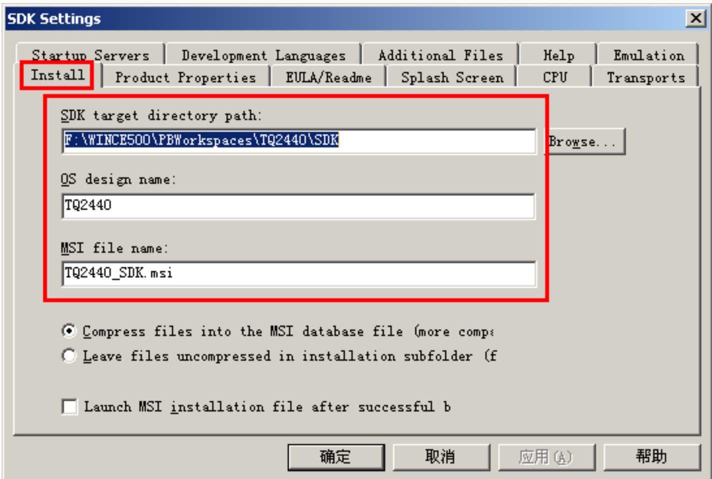
Click “Finish” to complete SDK Wizard:



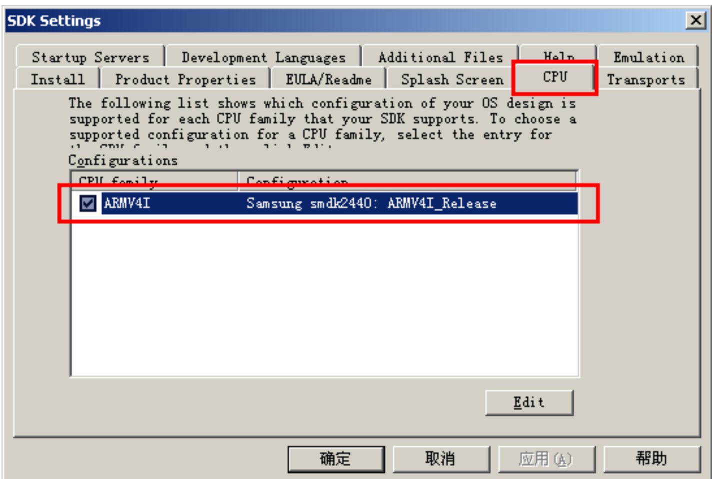
Select “Configure SDK” of “SDK” in the menu “Platform”:



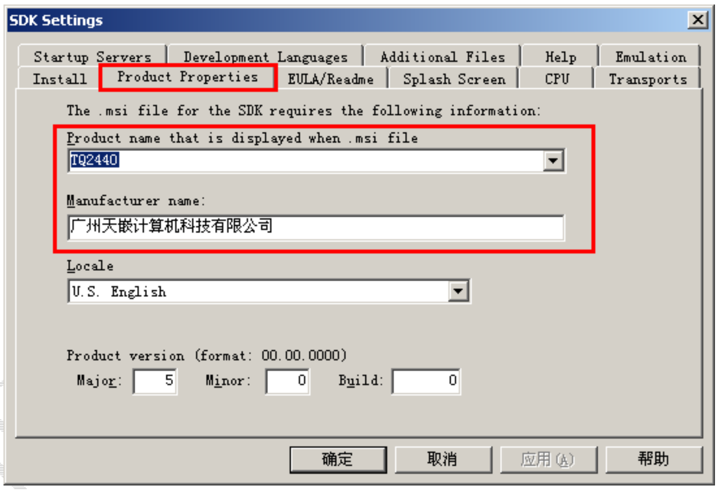
Configure the page “Install” in the interface “SDK Settings” as the following diagram:



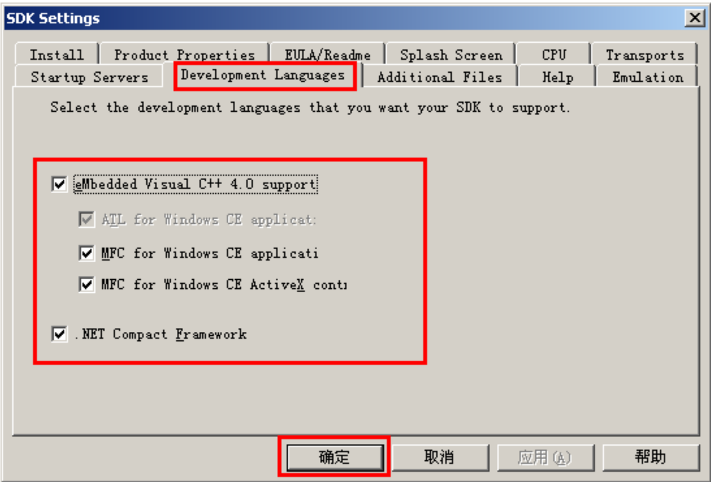
Configure the page “CPU” as the following diagram:



Configure the page “Product Properties” as the following diagram:



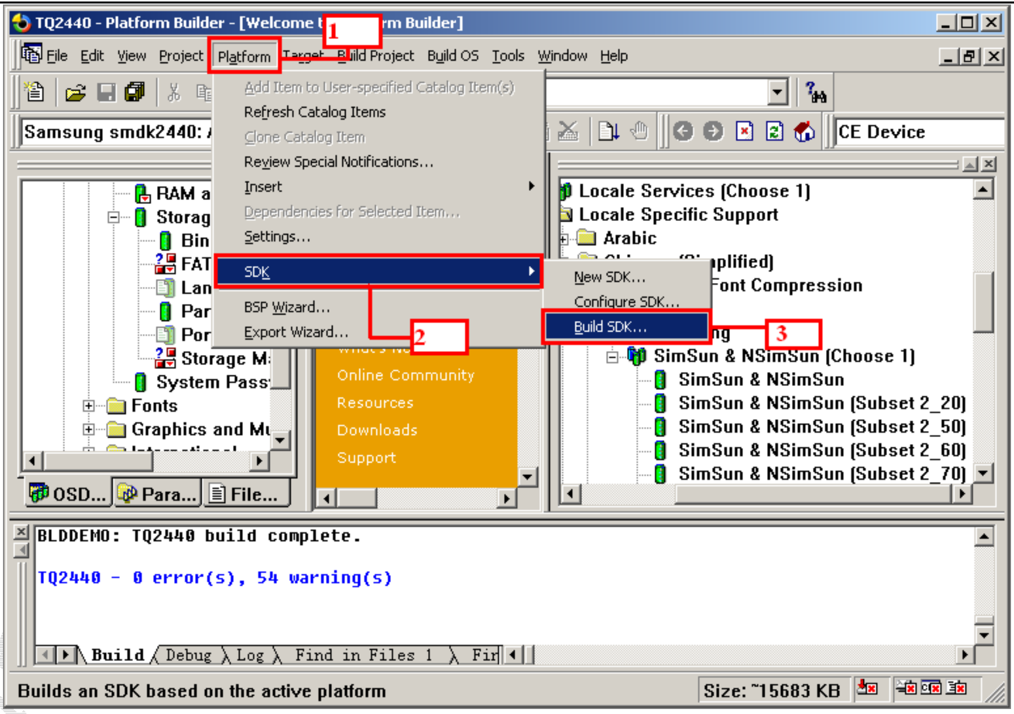
Configure the page “Development Languages” as the following diagram:



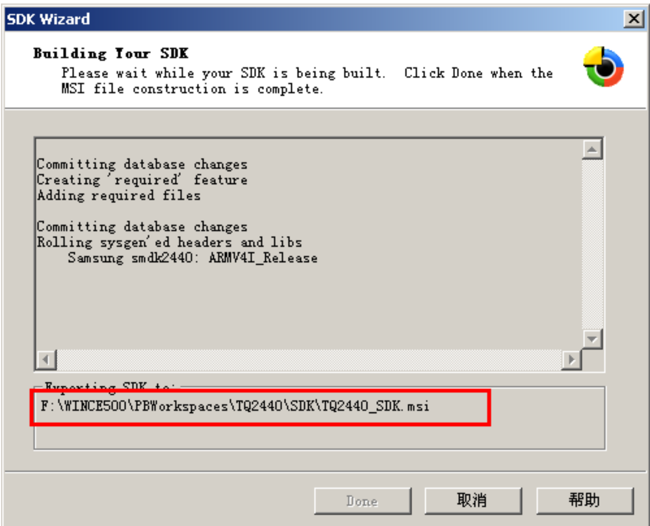
Click “确定” to finish the configuration.

5. 4. 2 Compiling SDK

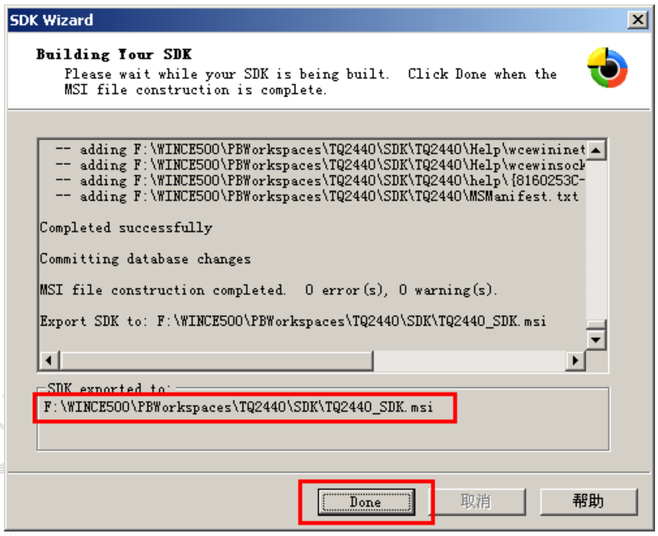
Select “Build SDK” of “SDK” in the menu “Platform”:



The following interface appears:

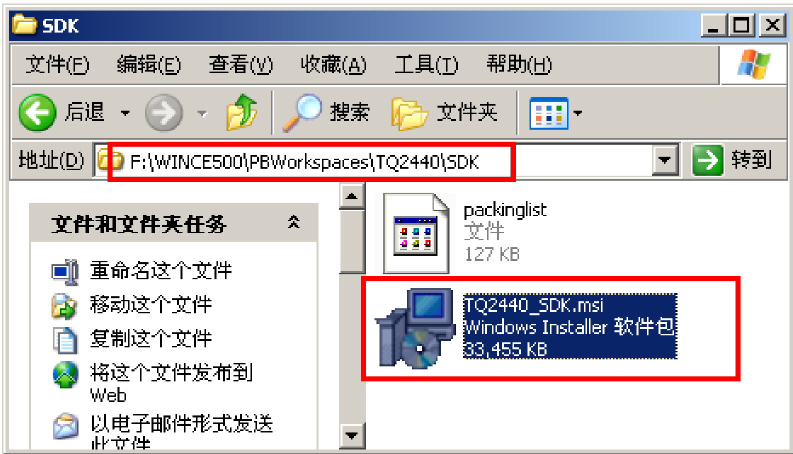


Click "Done" in the following diagram to finish compiling.



5. 4. 3 Finishing the compiling

The SDK installation package appears in the directory “F:\WINCE500\PBWorkspaces\TQ2440\SDK” after compiling:



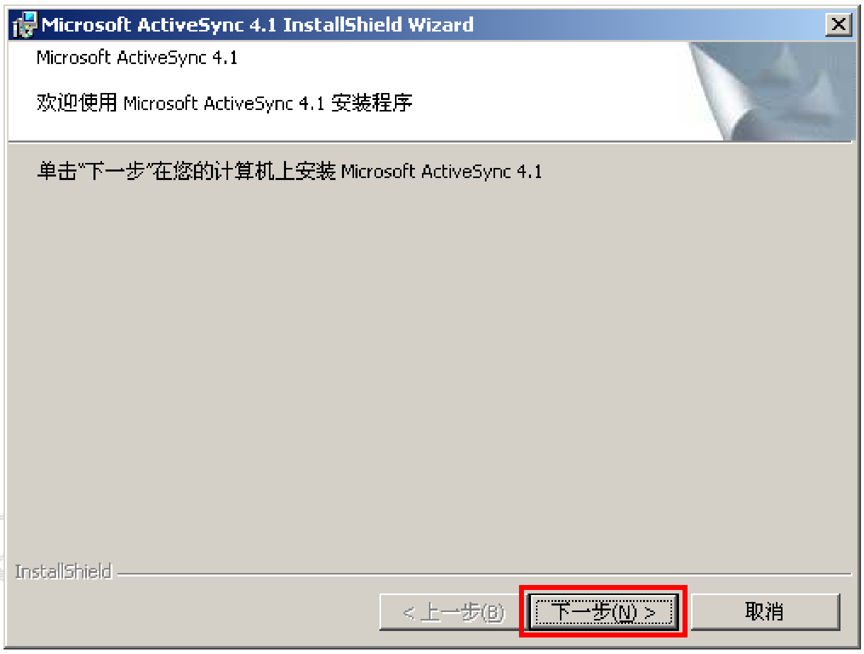
5. 5 Communicating with PC synchronously by using ActiveSync

The software “ActiveSync” can be used for communication between TQ2440 and PC, and it supports file transmission, remote debugging and other functions.

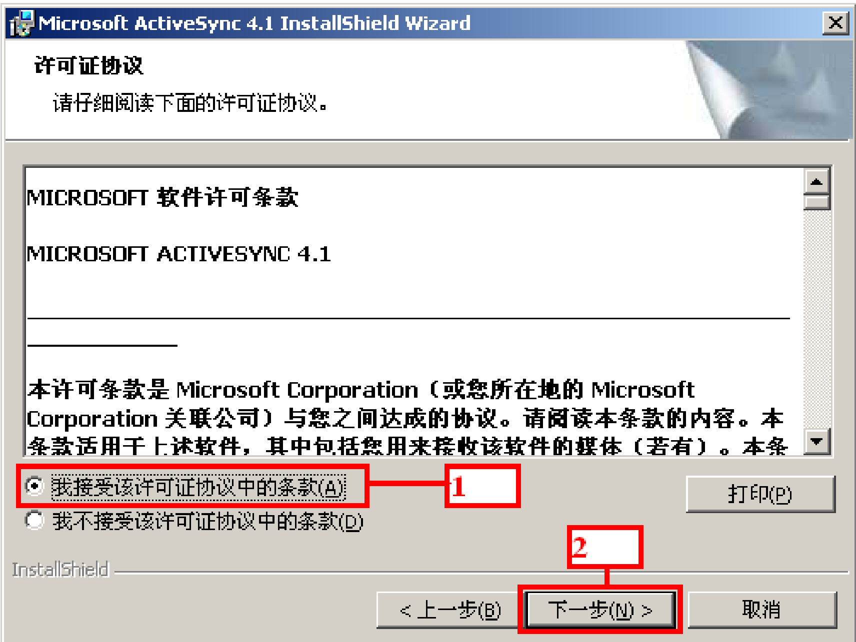
5. 5. 1 Installing ActiveSync

The ActiveSync installation application is under the directory “Windows 平台工具\ActiveSync” in CD-ROM.

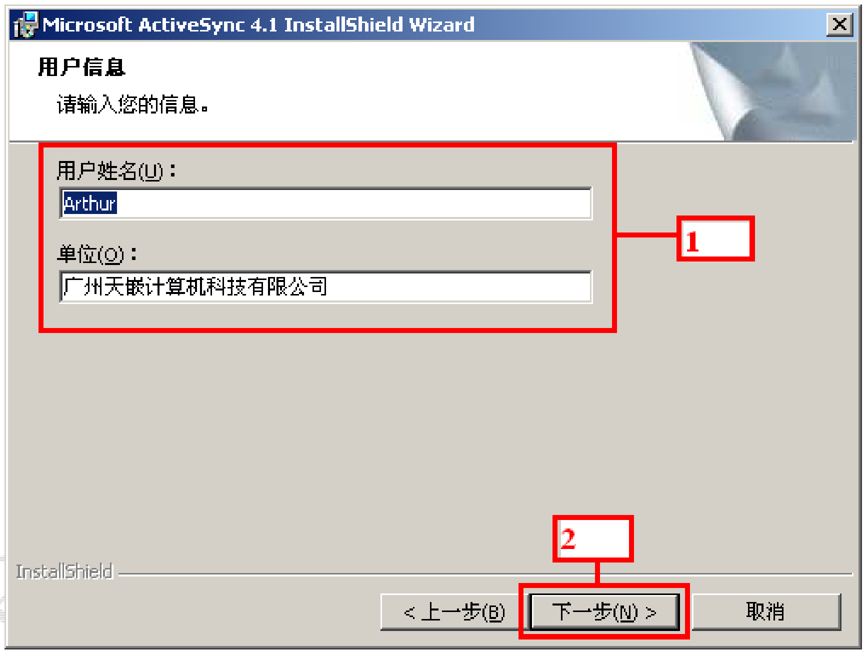
Step 1, double-click “ActiveSync_4.1_setup.exe” and click “下一步” to continue:



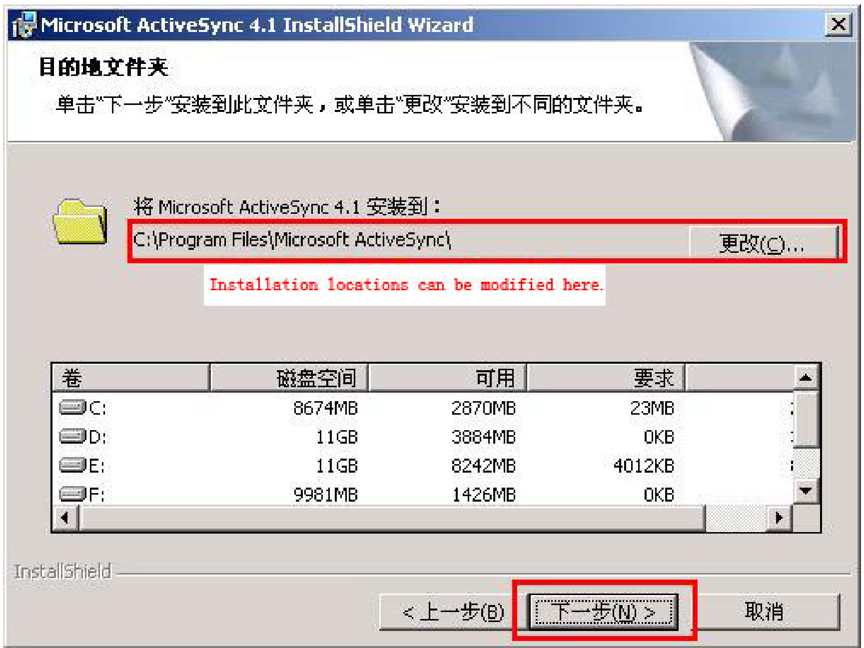
Step 2, select “我接受该许可协议中的条款” and click “下一步” to continue:



Step 3, enter the information “用户名” and “单位”, and click “下一步” to continue:

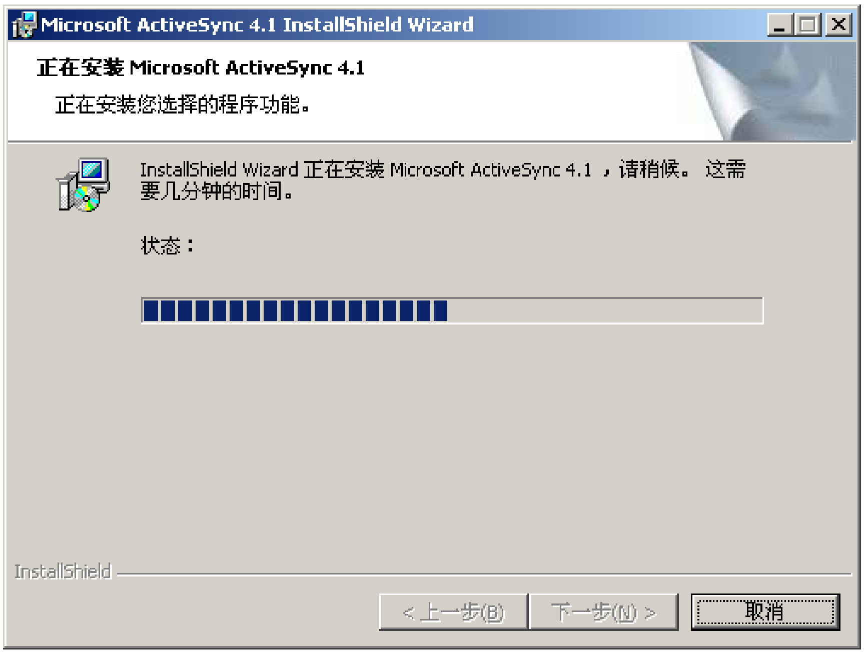


Step 4, select installation path “F:\Program Files\Microsoft ActiveSync\” and click “下一步” to continue:

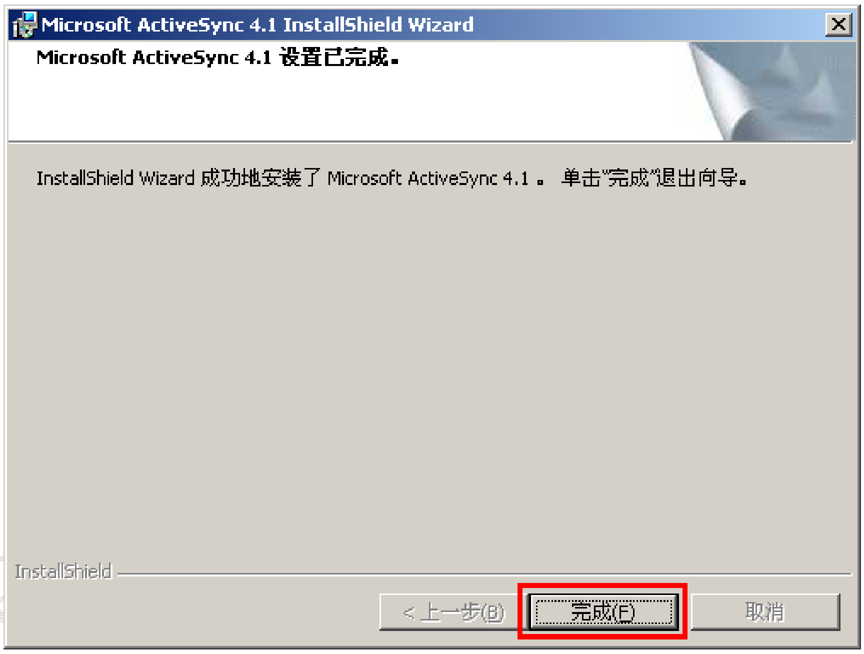


Step 5, click “安装” in the following interface to start installation:

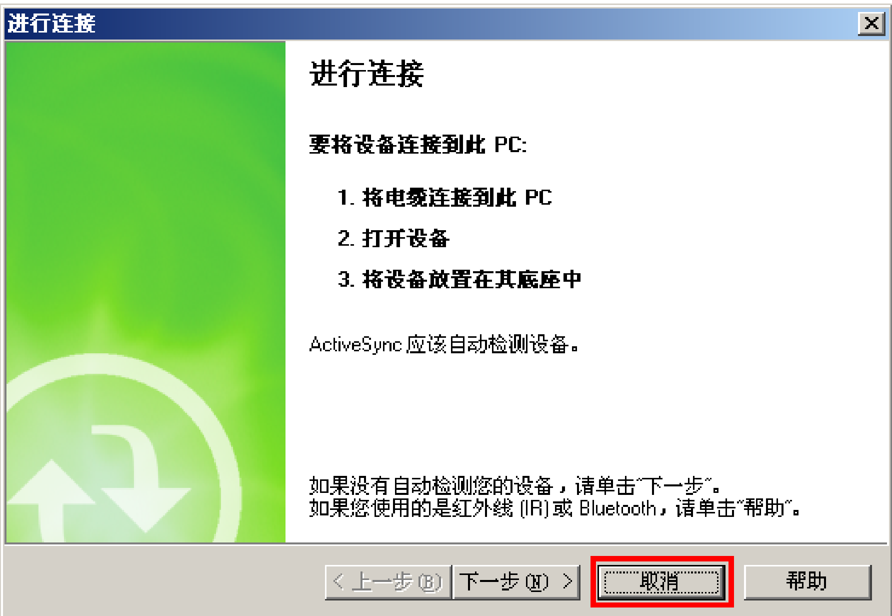
Installing:



Click “完成” to finish the installation:



The following interface appears:

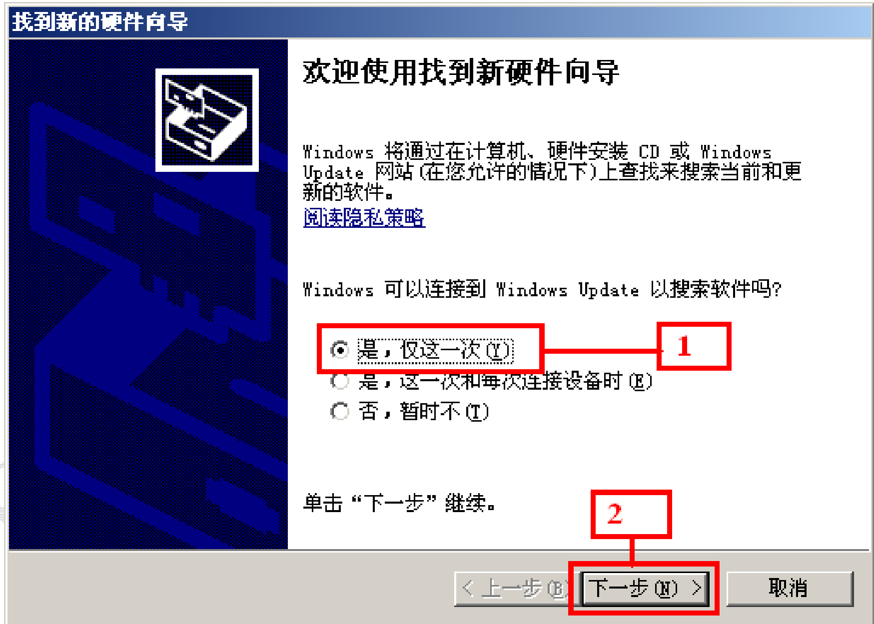


Click “取消”. The following icon appears in the taskbar:

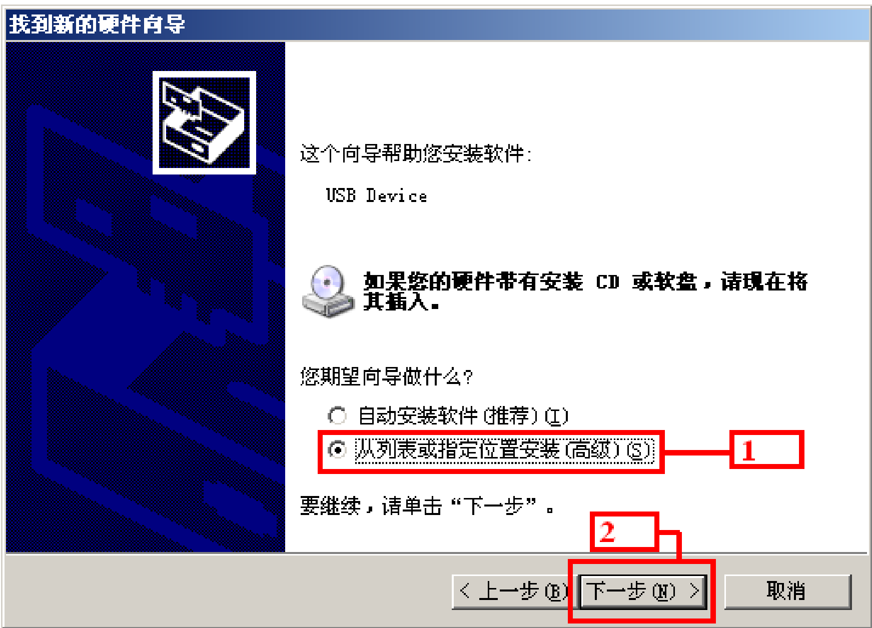
5. 5. 2 Installing synchronizing communication USB driver

Burn and start up WinCE image first. Connect PC and platform with USB wire. If there is no driver in PC, the prompt “发现新硬件” will appear, and you need to complete the installation according to the following steps. The driver installation program is under “WinCE\WinCE_4.2\SMDK2440\DRIVERS\USB\FUNCTION” of BSP package in CD-ROM.

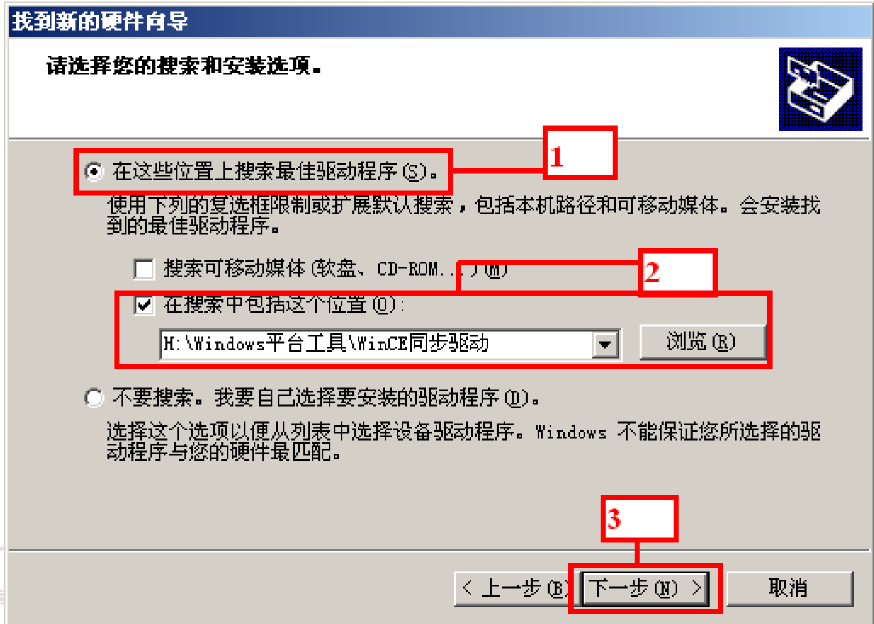
Step 1, the following interface appears after the USB wire has been connected. Select "是，仅这一次" and click "下一步" to continue:



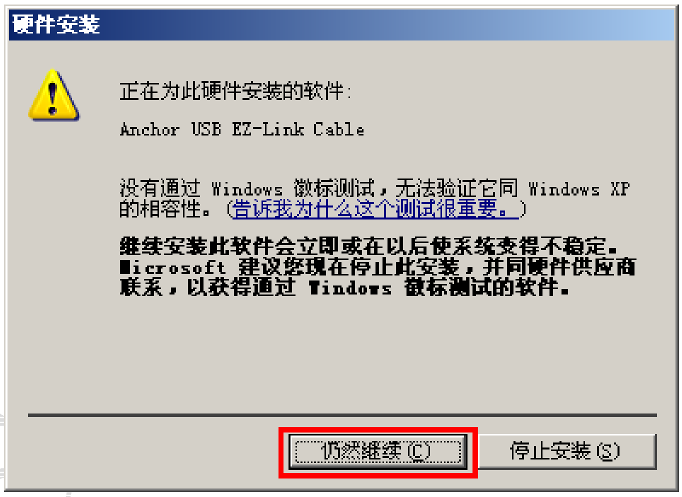
Step 2, select "从列表或指定位置安装(高级)" and click "下一步" to continue:



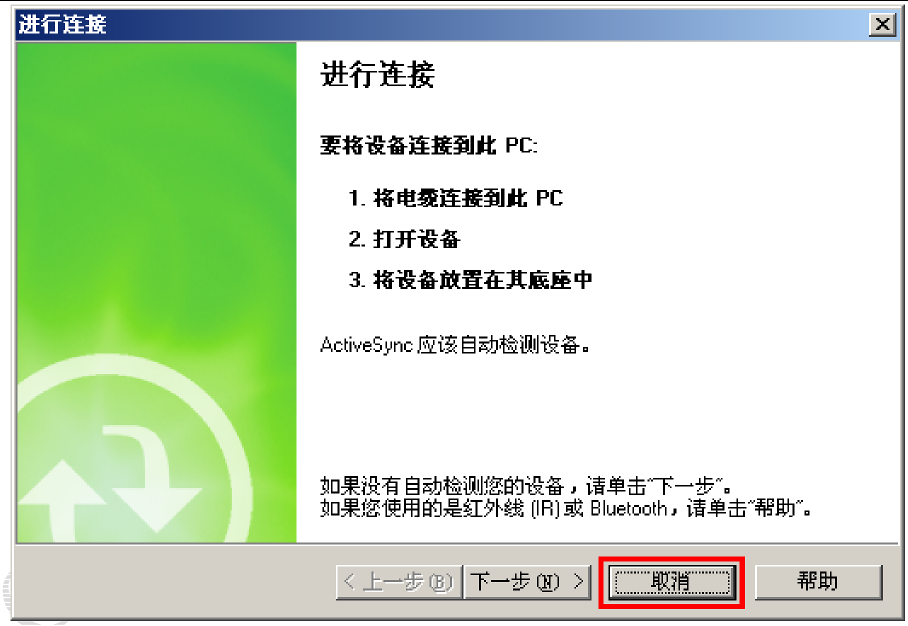
Step 3, select "在搜索中包含这个位置" and click "浏览" to locate the USB driver (under the directory "Windows 平台工具\WinCE 同步驱动" in CD-ROM). Click "下一步" to continue:



Step 4, the following diagram appear. Select “仍然继续” in the second interface and continue:



Step 5, USB driver installing is complete. Click “完成” and ActiveSync will run automatically.

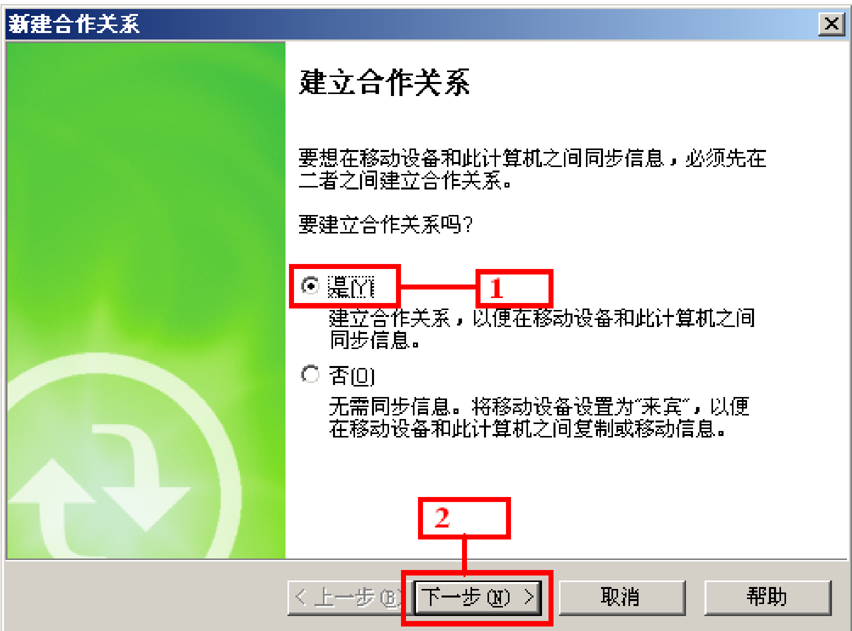


5. 5. 3 Utilize ActiveSync synchronizing software

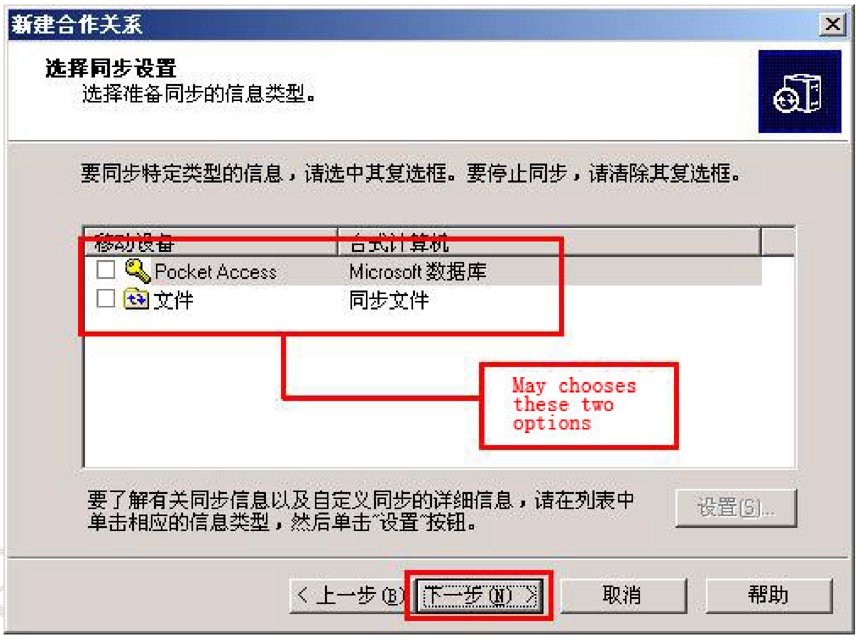
This part follows the upper steps.

Instruction: Make sure to remove all the U disk, SD card and other devices before using synchronizing function of WinCE.

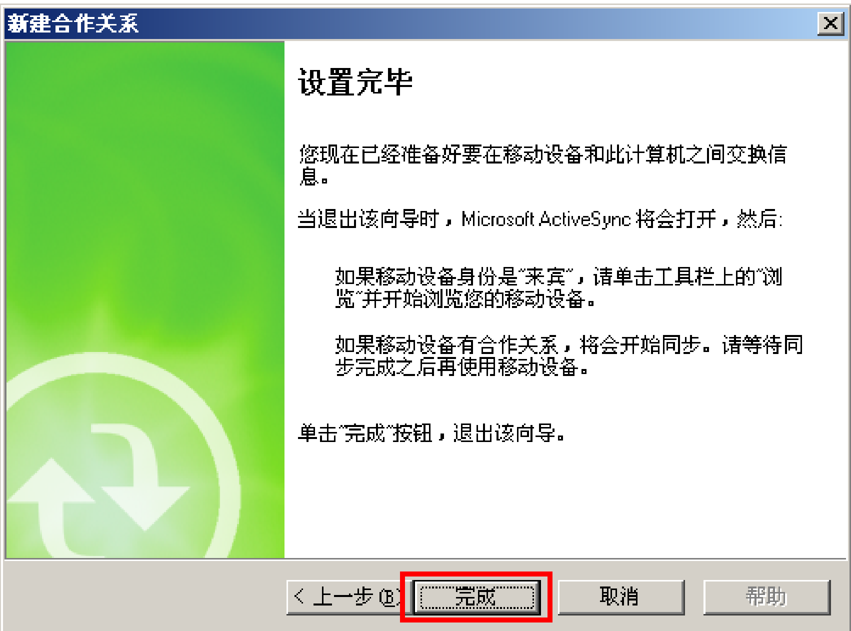
Step 1, select “是” in the interface “合作关系” and click “下一步” to continue:



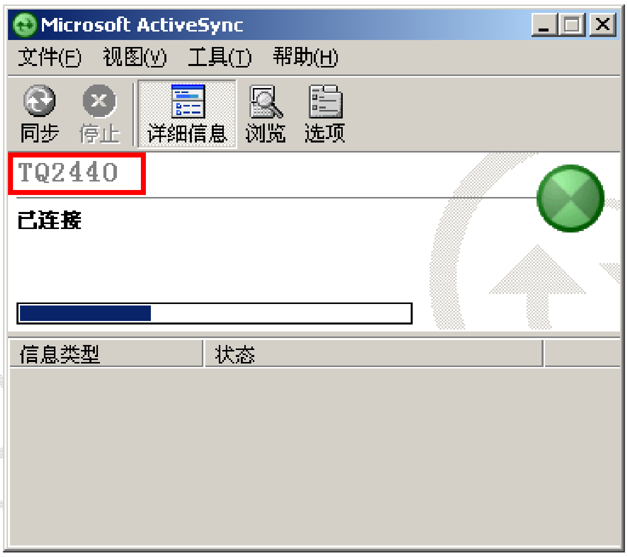
Step 2, click “下一步” in the interface “同步设置” and continue:



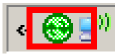
Step 3, click “完成” in the interface “设置完毕” to finish the configuration of ActiveSync



Step 4, the device connection interface “TQ2440” appears:

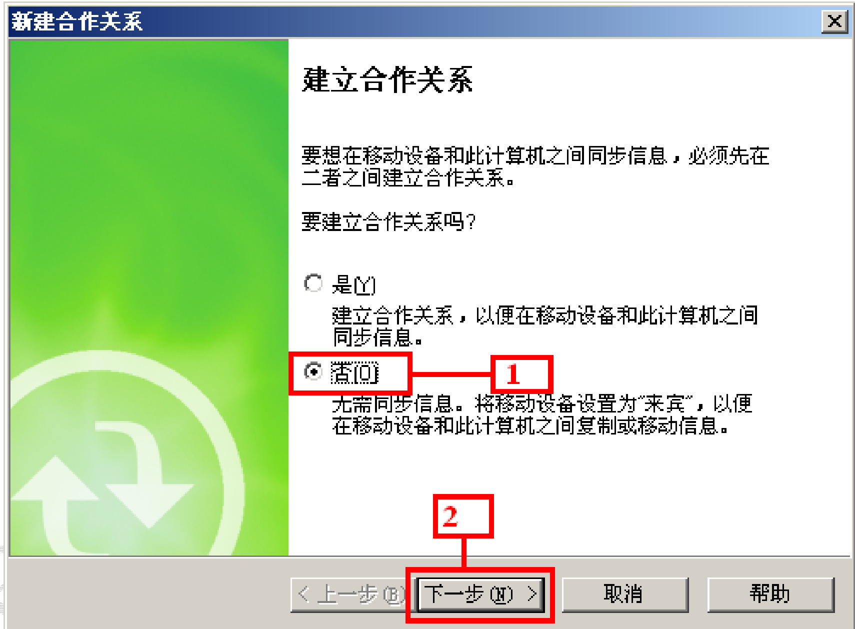


The synchronizing icon in the lower right corner turns green:

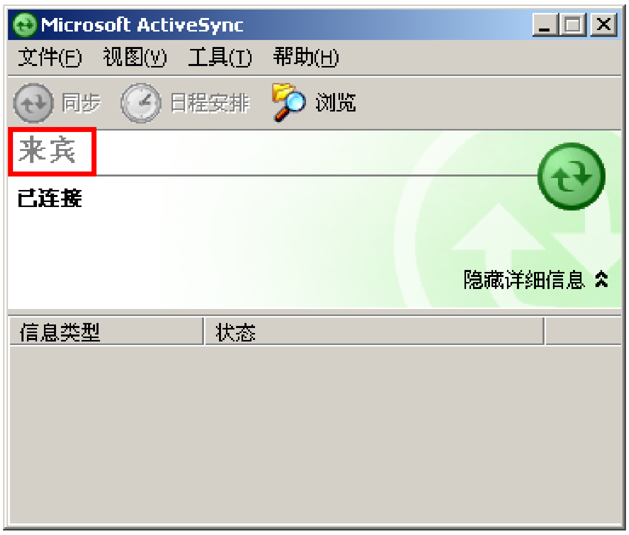


Step 5, the menu “移动设备” appears in “我的电脑”:

If select “否” in Step1:



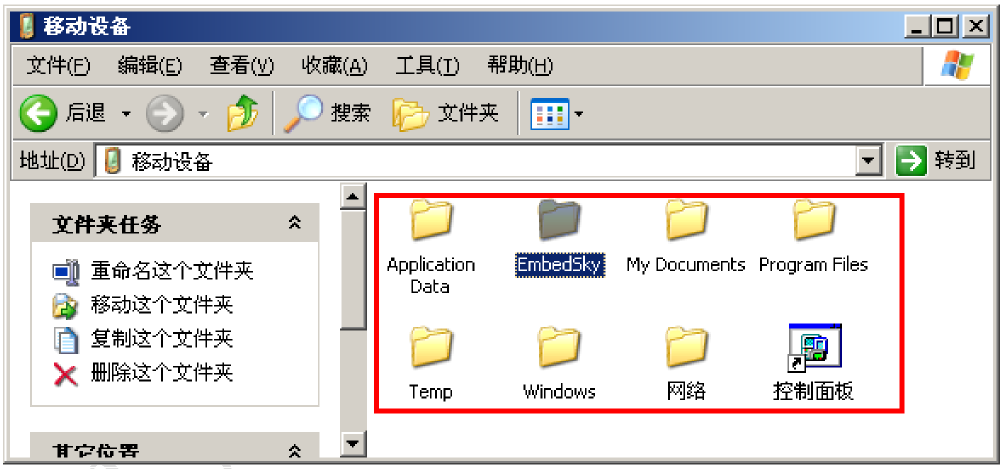
The connection interface “来宾” appears:



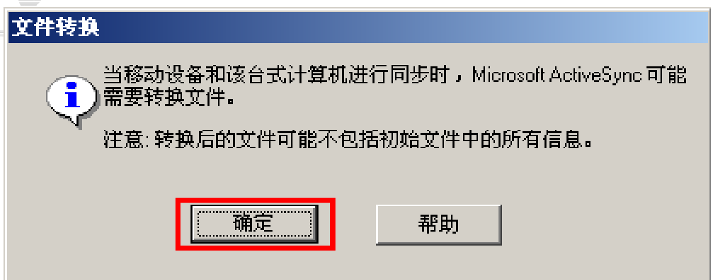
Instruction: the upper steps appear only in the first using. In the further using, the user need only modify the device name with other settings unchanged.

5. 5. 4 Transferring file with ActiveSync

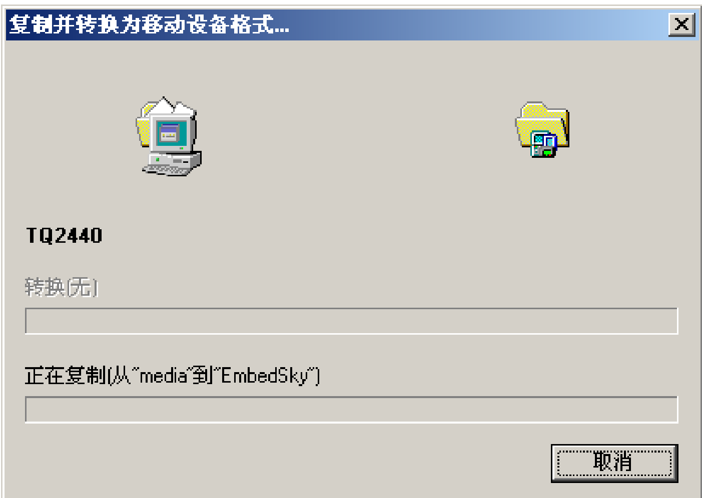
Step 1, after the previous operation, the platform has been synchronized with PC. Click “移动设备” and all directories in platform could be found:



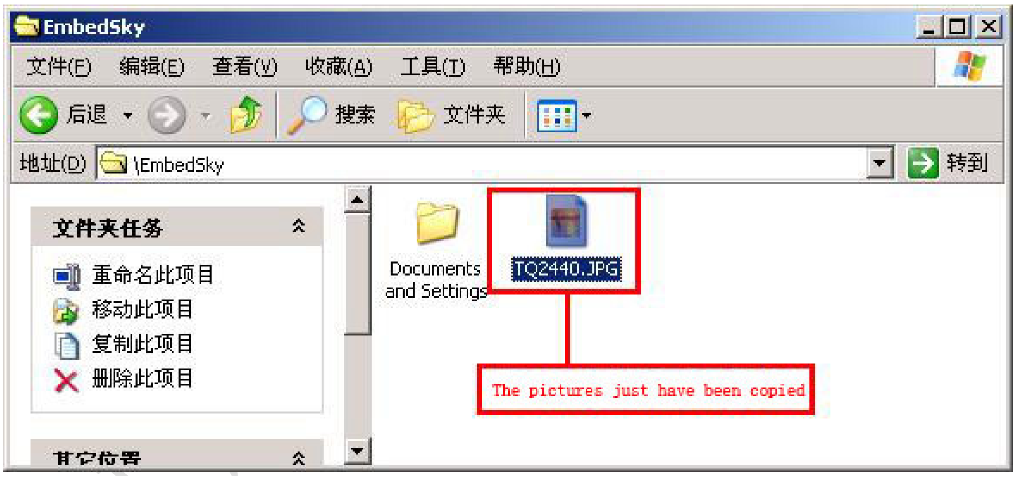
Step 2, copy a file from PC to the upper directory “EmbedSky”. The following interface appears:



Click “确定” to continue:

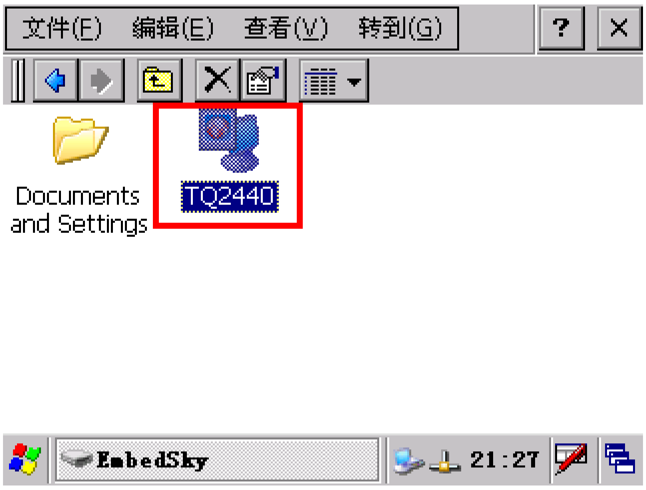


Step 3, after copying process complete, the file appears in the directory “EmbedSky”:



Step 4, the copied file could be found under the directory “EmbedSky” in “我的电脑”:

(instruction: The size of free space in Nand Flash is about 30M, please be aware of the free space.)



5.6 Capturing screen with ActiveSync and Platform Builder connection

Make sure the connection and synchronization between platform and PC is OK.

5.6.1 Configuring platform

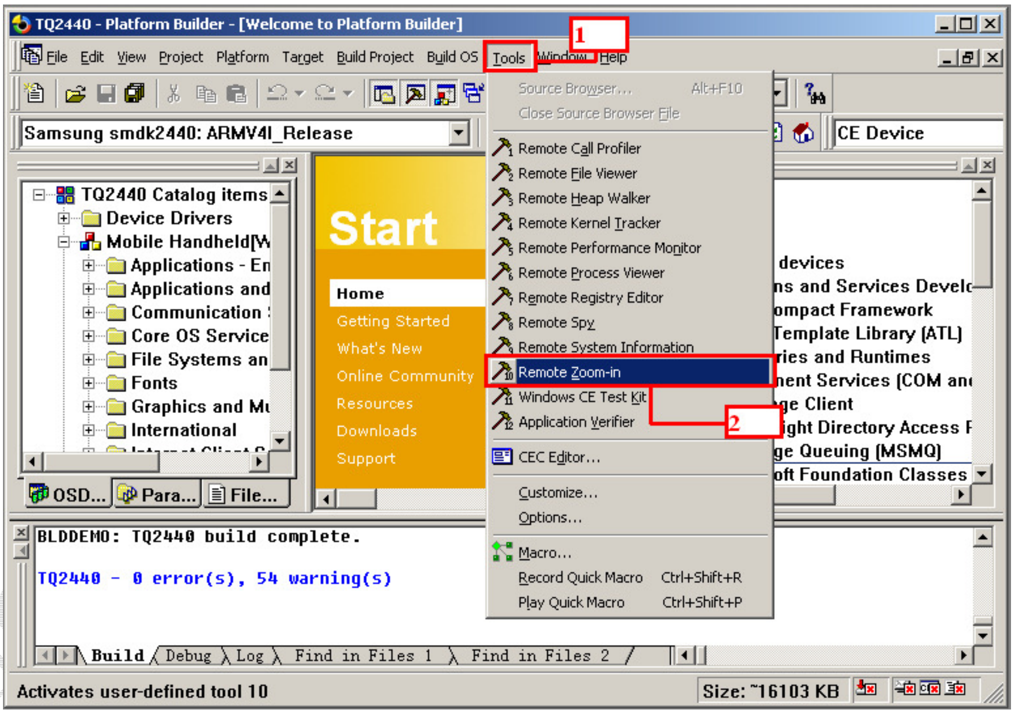
Connect USB wire and net line, and start up WinCE in platform.

If the platform doesn’t get connected to USB device, please restart the platform.

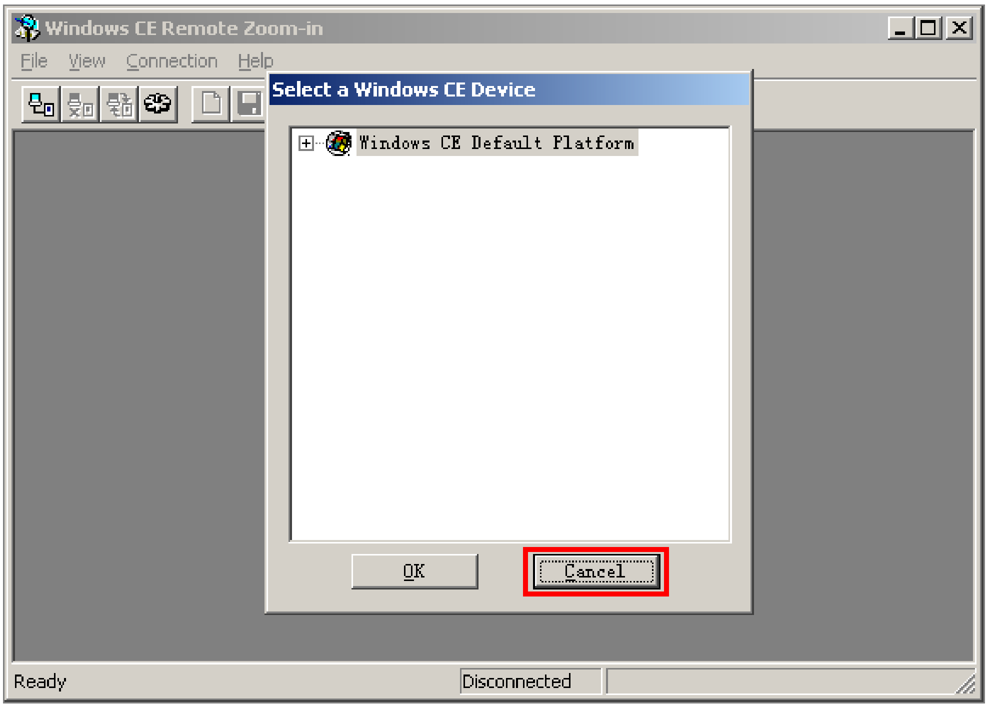
5.6.2 Configuring PC

Use the remote image scaling tools in PB.

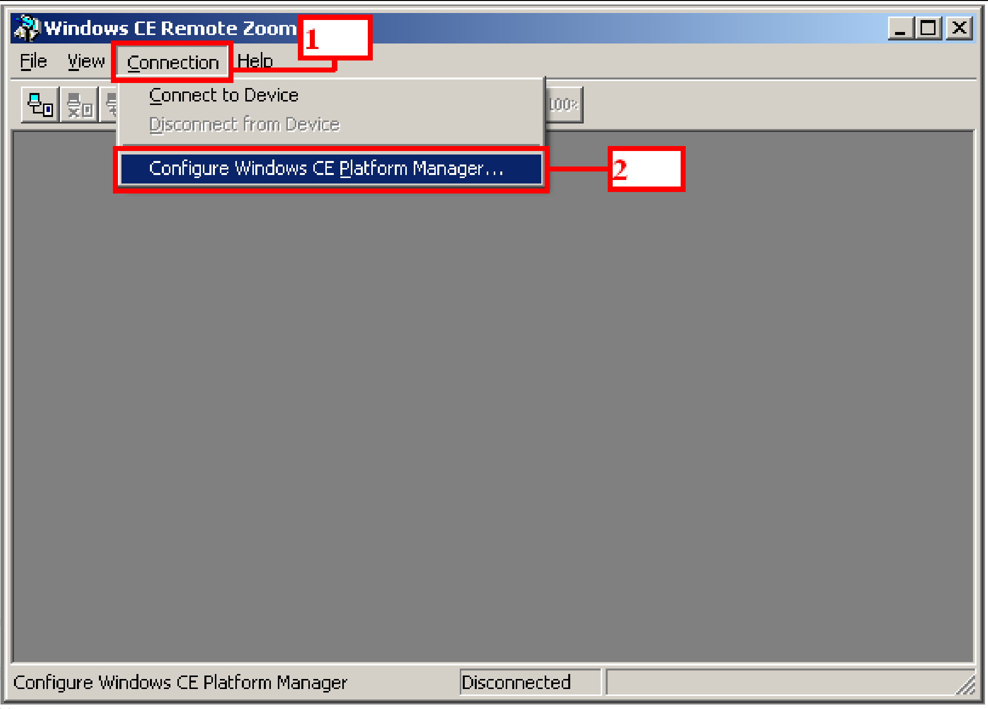
Step 1, click “Remote Zoom-in” in the menu “Tools” in PB:



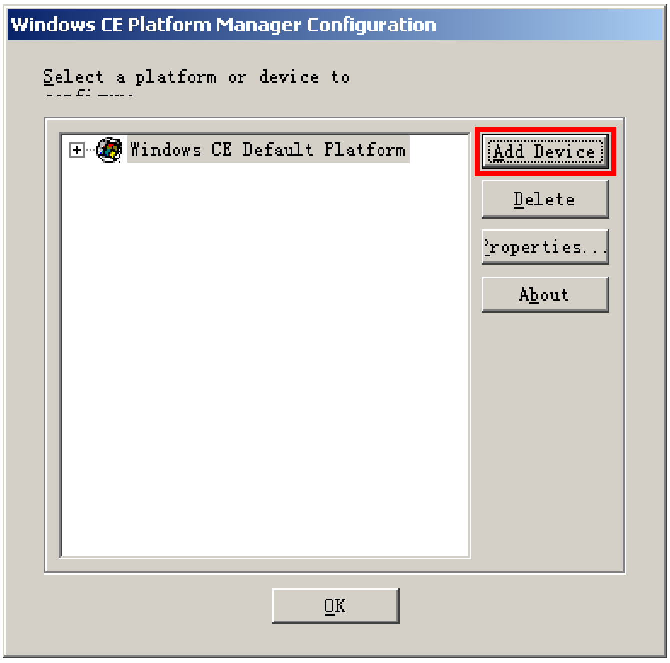
The interface of the software “Remote Zoom-in”:

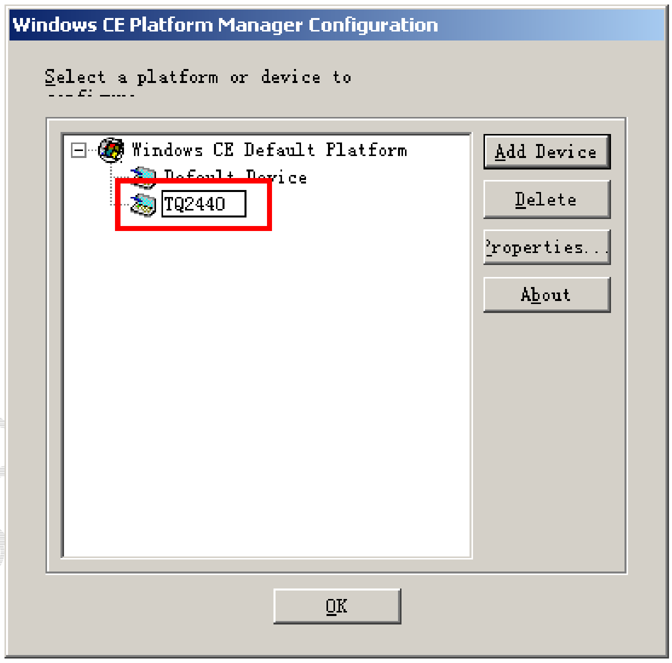


Step 2, click “Cancel” in the upper diagram, and click “Configure Windows CE Platform Manager” in the menu “Connection”:

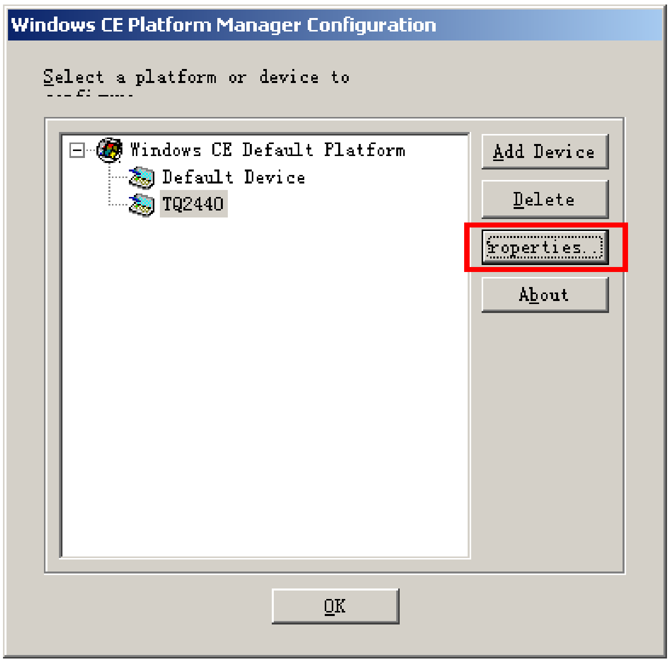


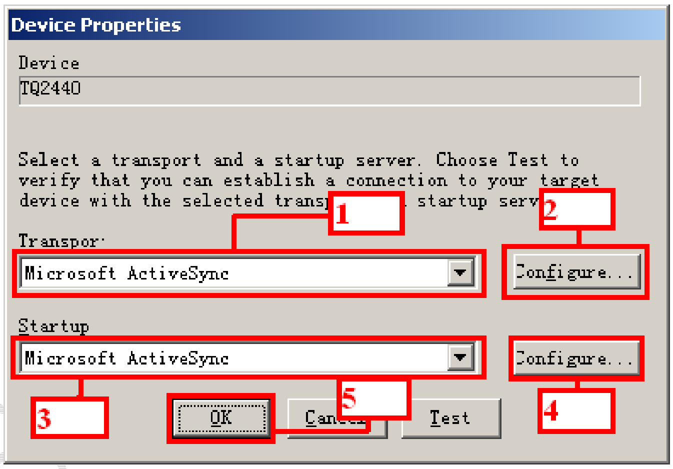
Select “Add Device” and add a new device named “TQ2440”.



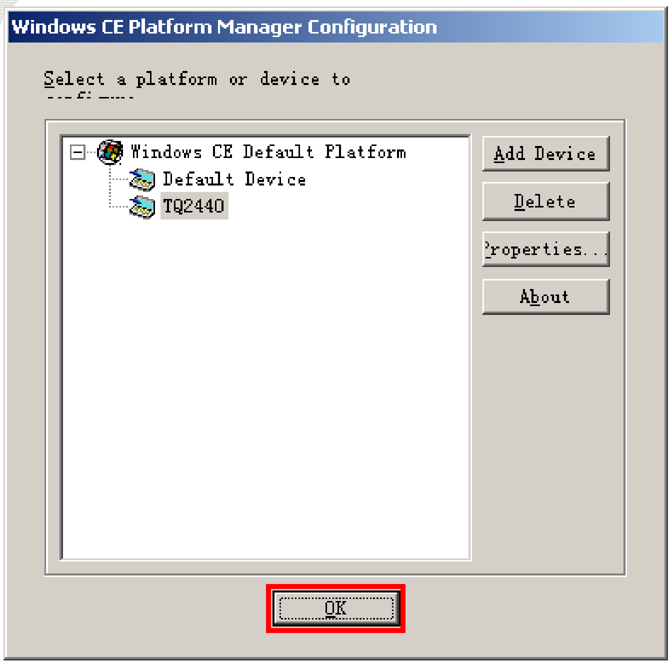


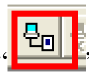
Click “Properties” or double-click “TQ2440”, and the configuration interface “Device Properties” appears. Select “Microsoft ActiveSync” in the option box “Transport”:

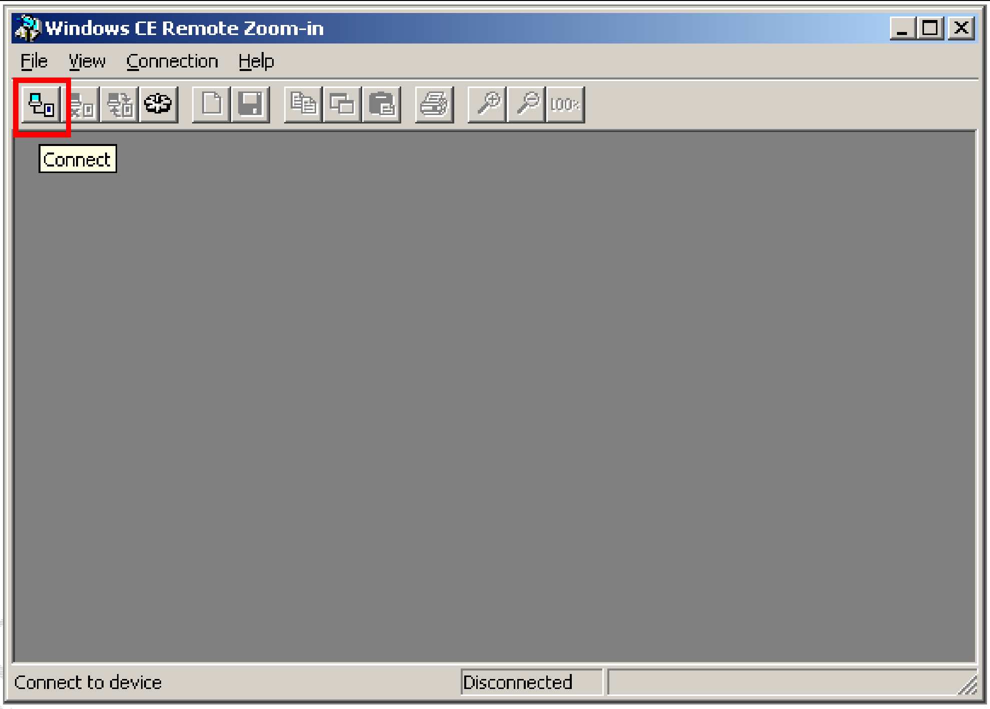




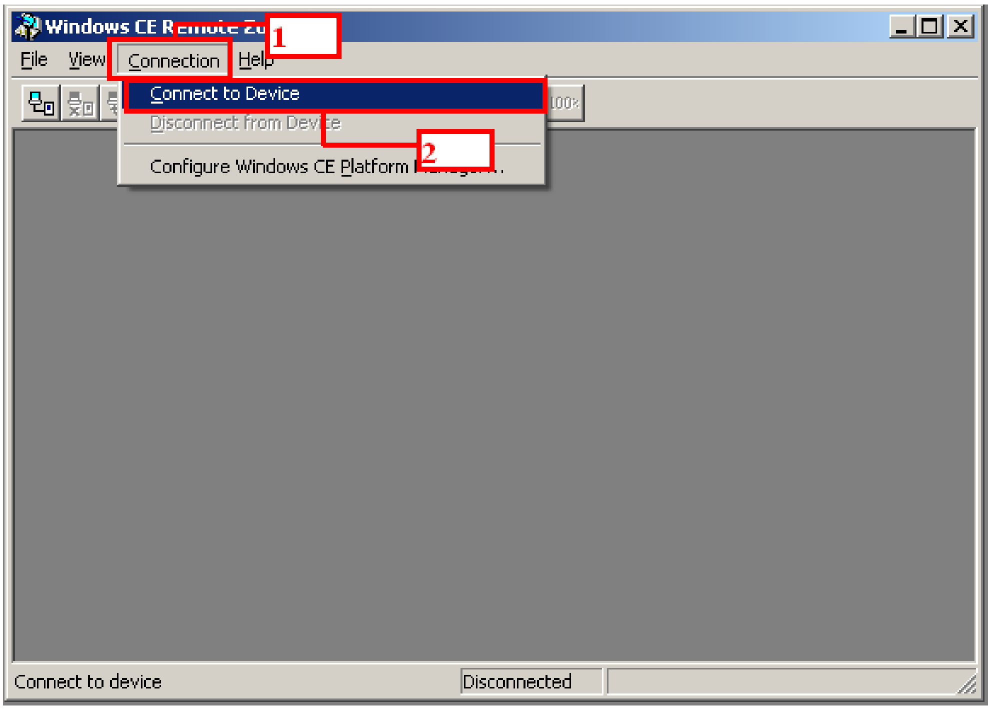
Click “OK” to finish the configuration:



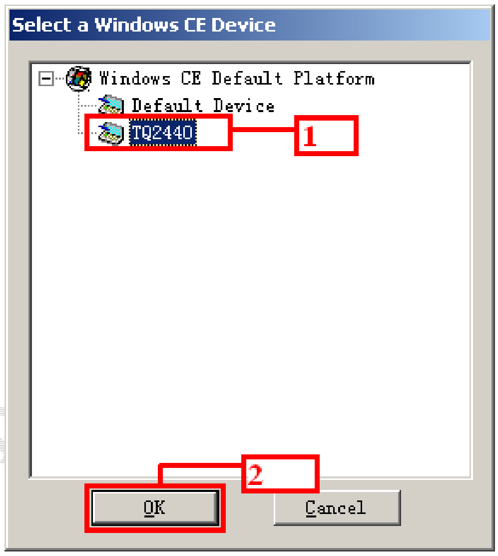
Step 3, click the icon “” in the upper left corner.



Or click “Connection->Connect to Device”:

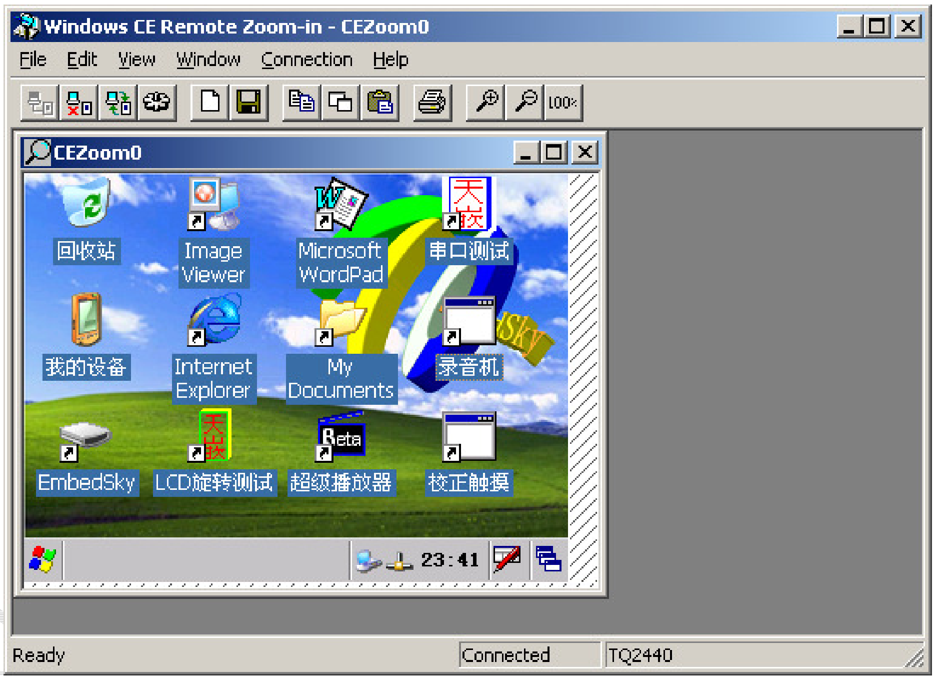


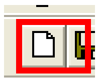
The following interface appears after clicking “connect”. Select “TQ2440” and click “OK” to continue:

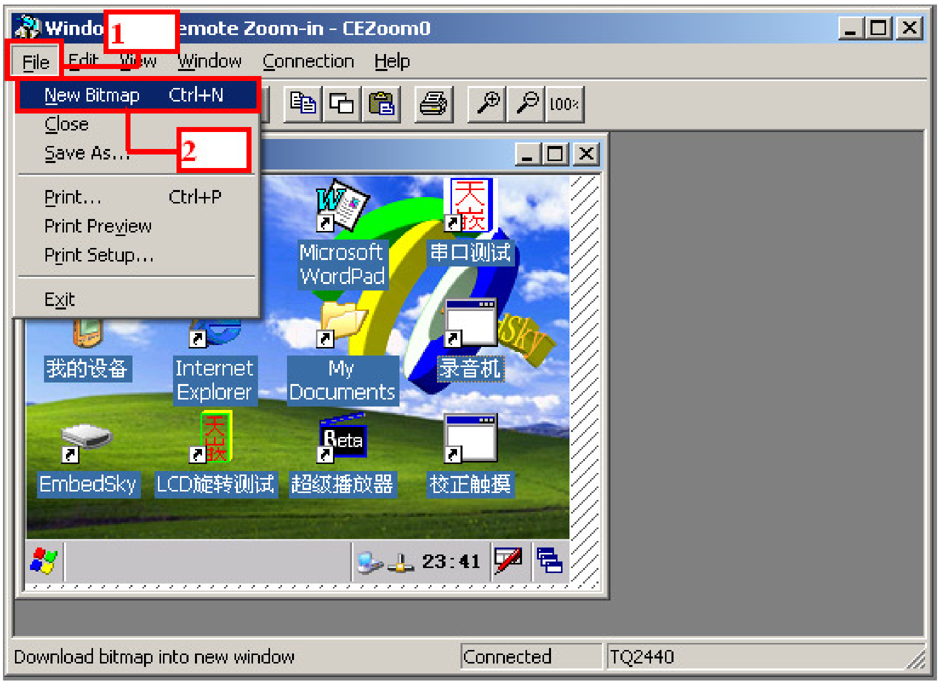


Step 4, PC and the platform start connecting after the upper operation (the net wire needs to be connected):

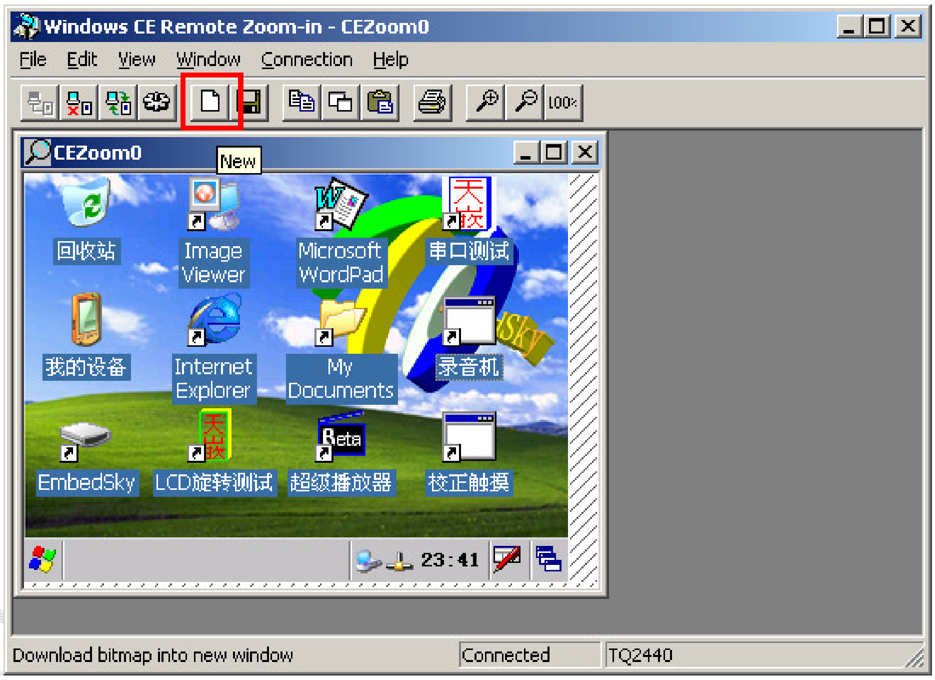
Connecting is complete:



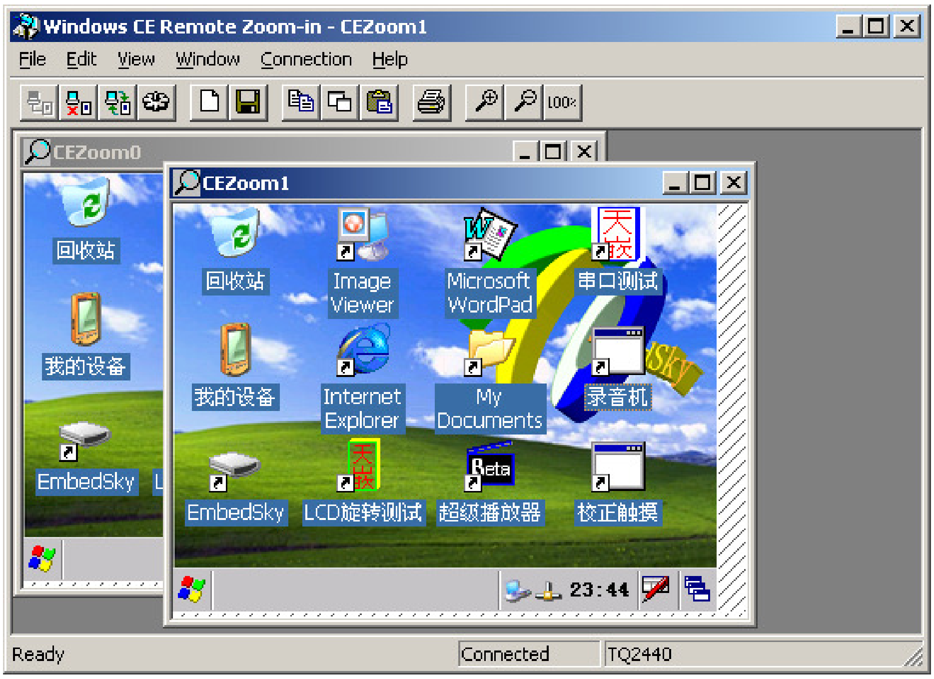
Step 5, click “New Bitmap” in the menu “File” or click  to get a new screen captured image:



Or:



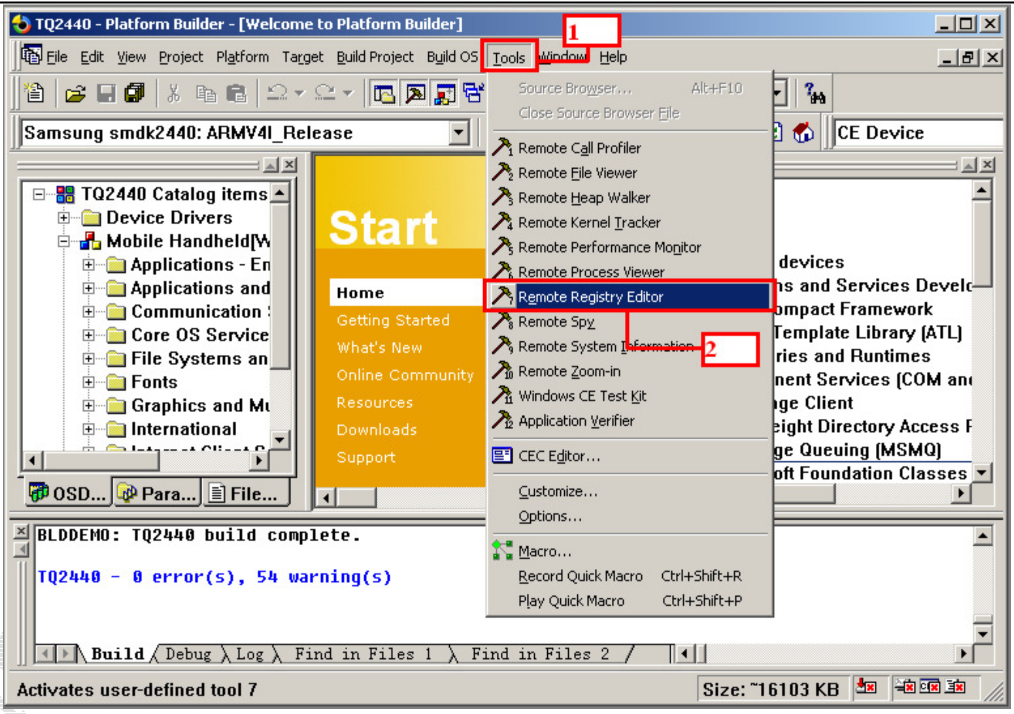
Get the second screen captured image:



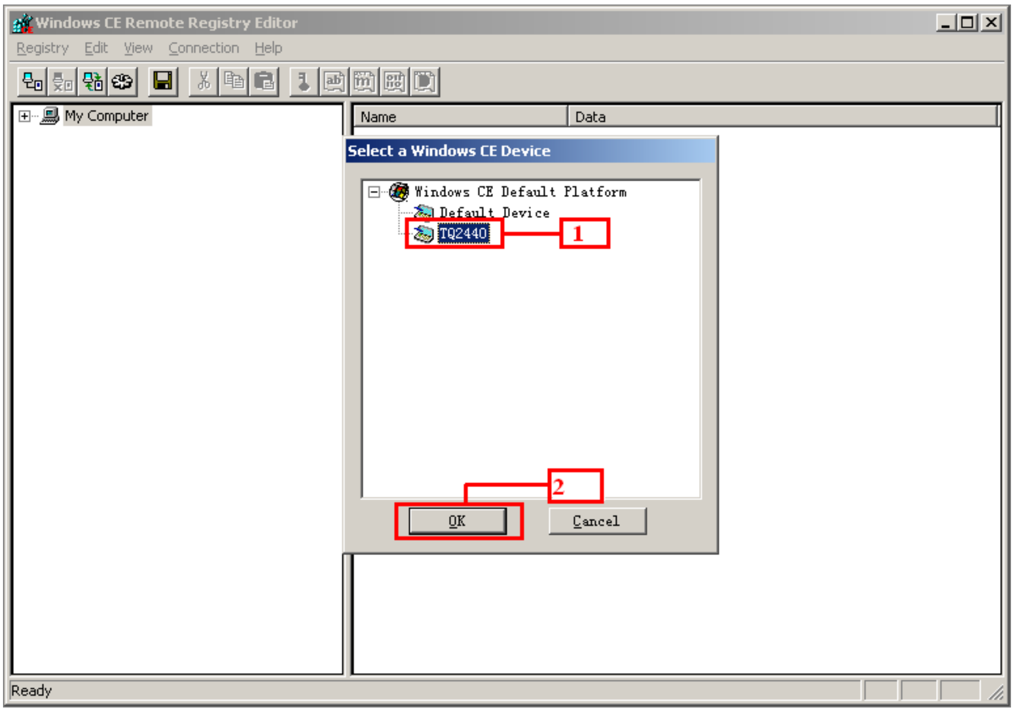
5.7 Check the platform register information based on connection between ActiveSync and Platform Builder

Make sure the connection and synchronization between platform and PC is OK.

Step1, open “Remote Registry Editor” in the menu “Tools” in PB and run remote register editor:

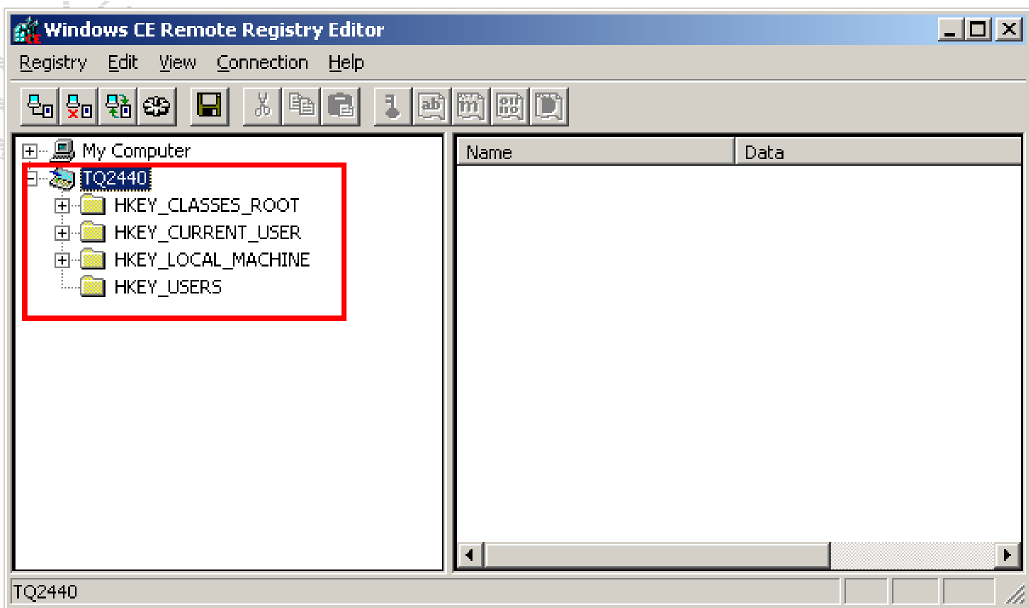


Step2, the following interface appears. All information has been set previously. Click “OK” to continue:

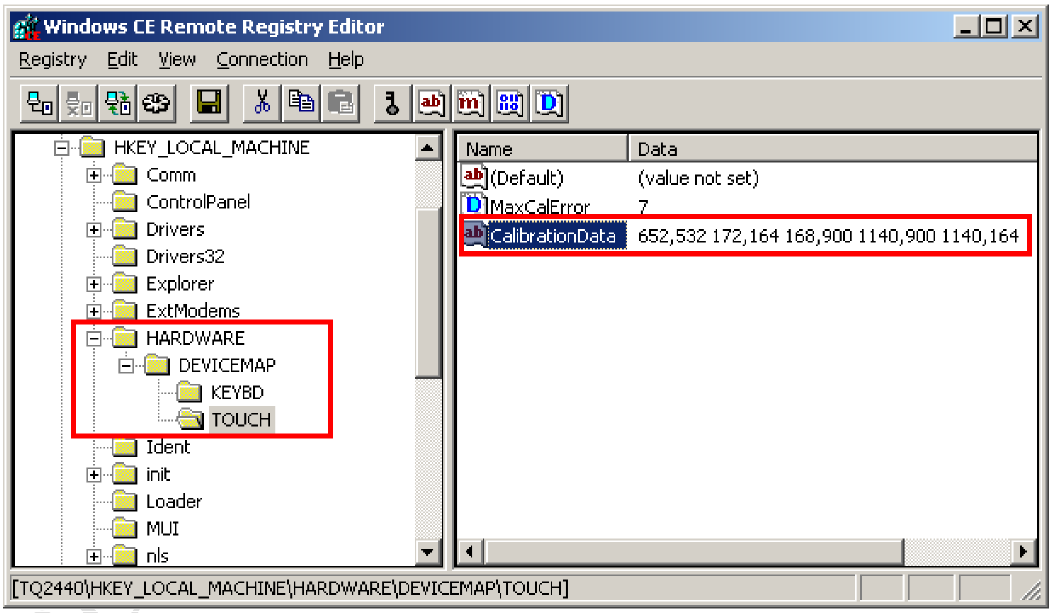


Make sure the net wire has been connected:

Step3, open the device file “TQ2440” after connection complete. The register is shown in the following diagram:



The touch screen register value:



What we introduce here is the basic operation of WinCE development. There are a lot more applications about PB and platform connection, which will introduced in the following examples.

Chapter 6 Qt/Embeded graphics development

Qt is application development frame supporting multi-Operating System. The development language is C++. Qt provides a unified and exquisite graphics program interface and a unified network and database operation program interface.

Qt is given to the developer in the form of SDK (software development kit), including graphics designer, Makefile designer, font internationalization tool and cross-platform C++ class libraries

Qt resources:

Trolltech homepage: http://www.trolltech.com

FTP supporting anonymous access： ftp://ftp.trolltech.com

Newsgroup server: nntp.trolltech.com

Non-official Qt document Chinese translation group: http://www.qiliang.net/qt/index.html

Qt/Embeded is a C++ SDK customized for user graphics interface and application development. It supports embedded Linux on different processors.

The resources for building Qt/Embeded development environment:

- Tmake kit: tmake-1.11.tar.gz
  - — Used for creating project Makefile;
- Qt/Embeded installer: qt-embeded-2.3.7.tar.gz
  - — Used for installing Qt/Embeded;
- X11 version installer of Qt: qt-x11-2.3.2.tar.gz
  - — Used for creating some necessary tools;
- Qtopia installer: qtopia-free-1.7.0.tar.gz
  - — Provide graphics interface for handheld device;
- Script build and script setenv
  - — They are respectively compiling script and path configuration script.

There is a rule for selecting the tool package: the version of Qt for X11 needs to be older than Qt/Embeded. Because the source file created by the tools "uic" and "designer" of Qt for X11 installer will be compiled together with Qt/Embeded library. A more updated version of Qt for X11 might encounter compatibility problem with Qt/Embeded library.

Qt/Embeded source code package is namely the file "Qte.tar.bz2" under the directory "Linux \Linux-2.6.13\" in CD-ROM. After decompression, the file is located at:

- X86 version: /opt/EmbedSky/Qte/x86-qtopia/
- The ARM version supporting USB mouse and keyboard: /opt/EmbedSky/Qte /mouse-qtopia/
- The ARM version supporting touch screen: /opt/EmbedSky/Qte /touch-qtopia/

(caution: the upper 3 versions get different compiling scripts, but their source codes are the same. What's more, please use the cross-compiler version 3.3.2)

6. 1 Simulating Qt/Embeded in PC

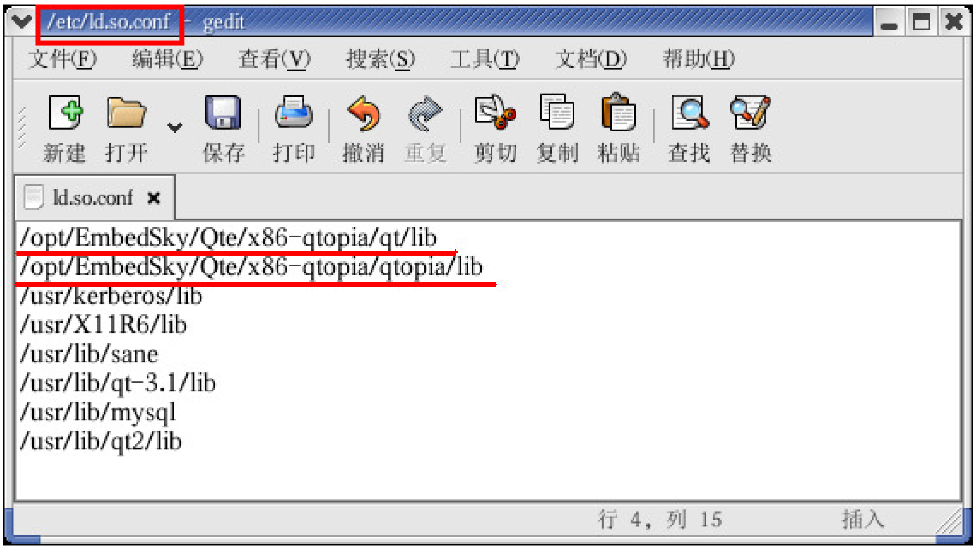
6. 1. 1 Configuring the running evironment

The Qt/Embeded simulation in PC needs the support of Qt/Embeded library files. The user needs to modify the file "/etc/ld.so.conf" to fit it to Qt/Embeded development platform. (although Redhat includes Qt library, the version might not be suitable.)

#gedit /etc/ld.so.conf //modify the contents as follows:

```
/opt/EmbedSky/Qte/x86-qtopia/qt/lib
/opt/EmbedSky/Qte/x86-qtopia/qtopia/lib
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/sane
/usr/lib/qt-3.1/lib
/usr/lib/mysql
/usr/lib/qt2/lib
```

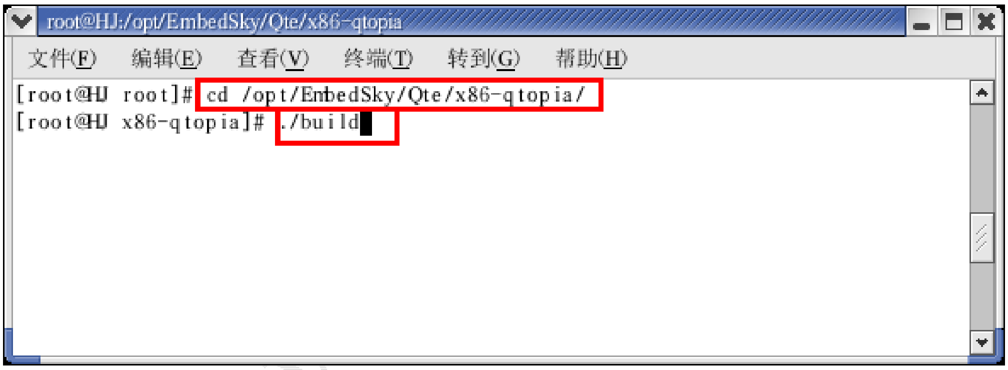
Or as the following diagram:



5. 1. 2 Installation and compiling

Executing the following commands to install and compile the tool scripts provided by us:

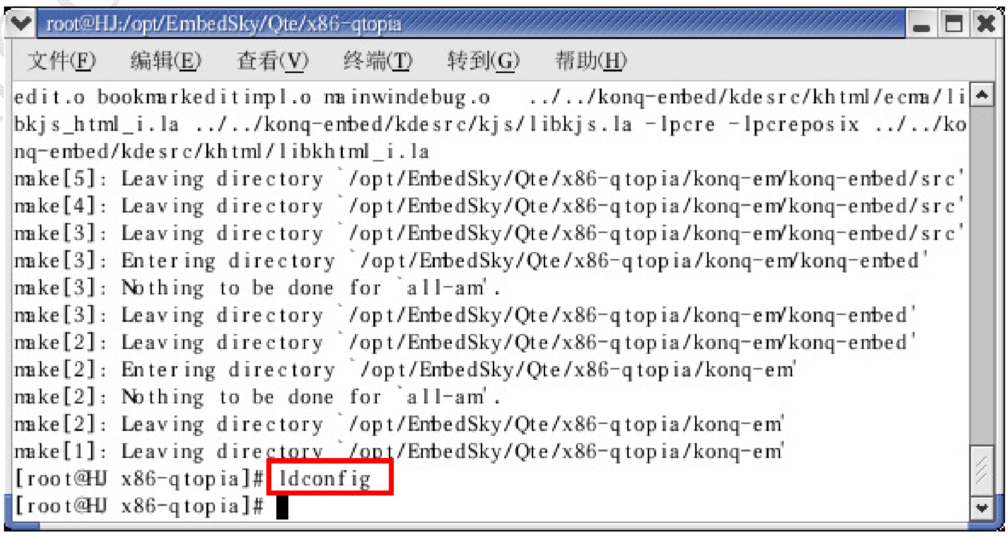
```
#cd /opt/EmbedSky/Qte/x86-qtopia/
#./build
```



It is needed about 20 minutes for installing and compiling.

5. 1. 3 Enable the created library

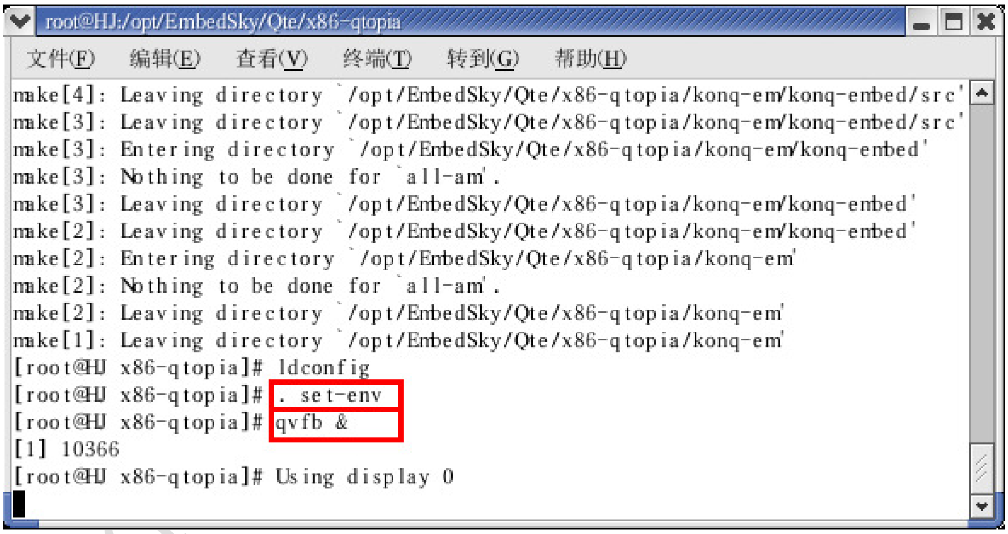
#ldconfig //Enable the modified file "/etc/ld.so.conf".



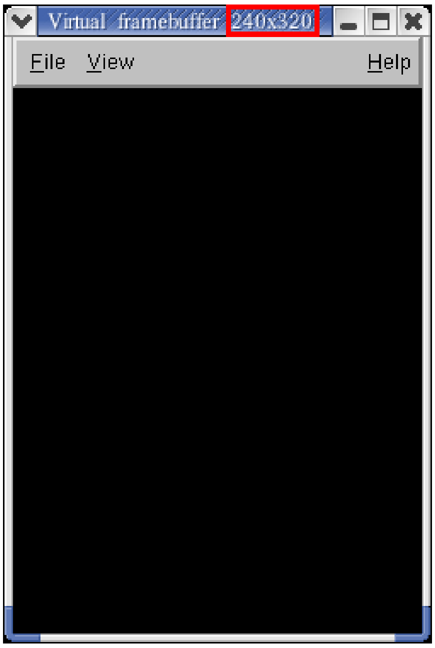
5. 1. 4 Simulating Qtopia in PC

#. set-env //Configuring environment parameters to support Qtopia simulation. (caution: The blank character between "." and "set-env" is indispensable!)

#qvfb & //The default display size is 240×320.

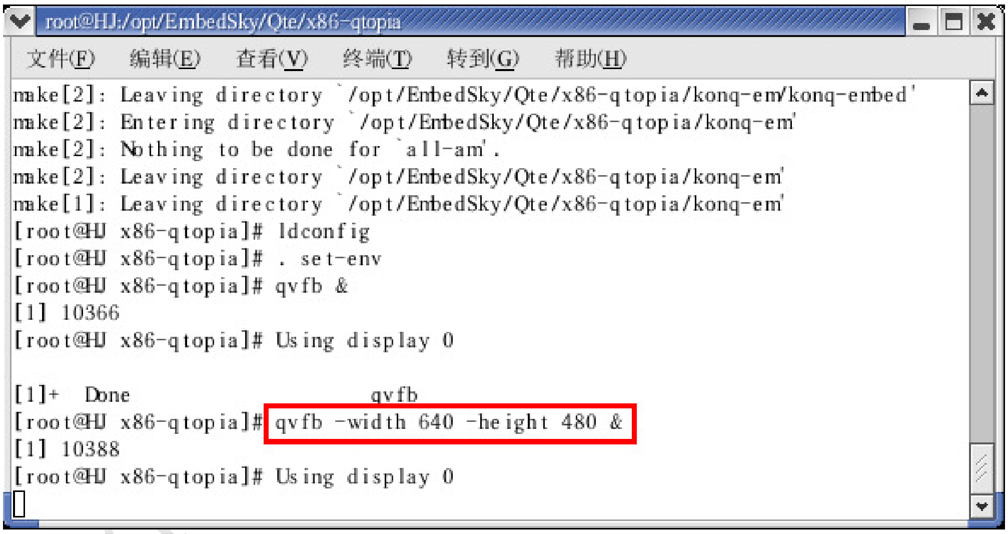


The running interface of “qvfb”:

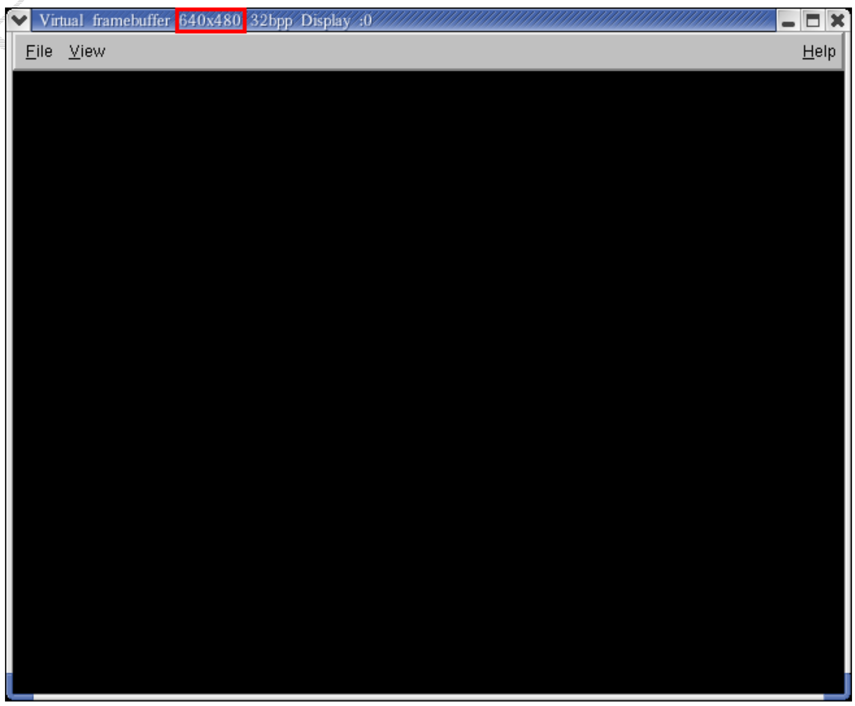


Enter the following command to modify the display size to 640×480:

#qvfb –width 640 –height 480 &

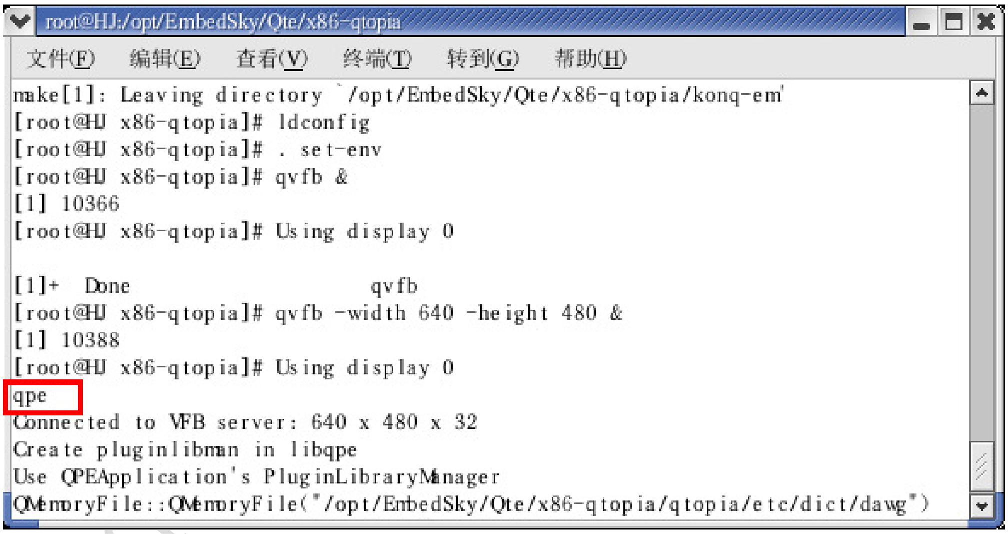


The running interface:



Execute the command:

#qpe



The simulation interface:

6. 2 Programming

6. 2. 1 Hello routine

Find the routine “hello” under the directory “/opt/TQ/Qt/x86/”, or write a program referring to the routine “Hello”. (this part of chapter does not introduce how to write a program, but how to compile routine.)

6. 2. 2 Configuring environment parameters

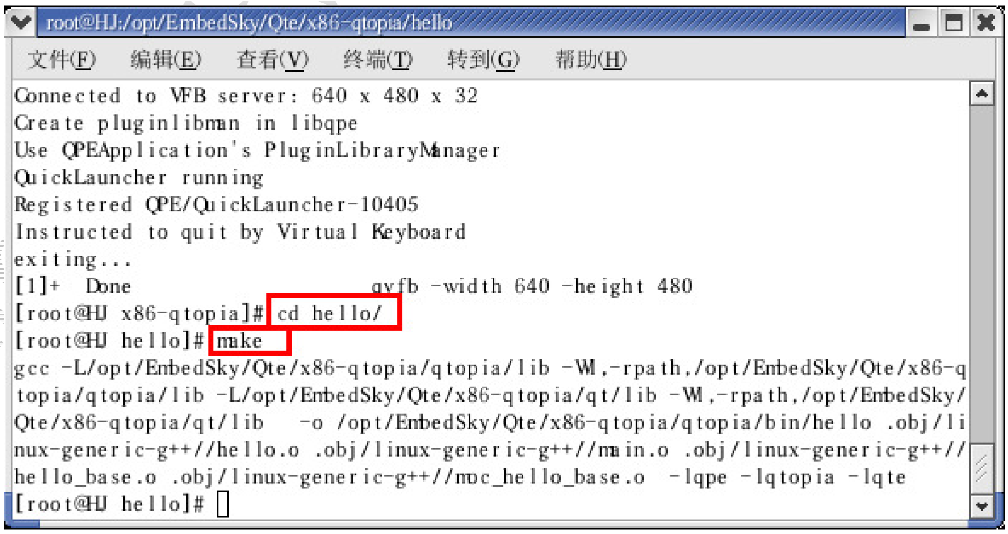
Execute the script “set_env”. (caution: Complete the steps in “5.1.2” and “5.1.3” before compiling the routine “Hello”.)

#cd /opt/EmbedSky/Qte/x86-qtopia/

#. set-env (caution: The blank character between “.” and “set-env” is indispensable!)

#cd hello

#make



The executable file “hello” is created under the directory “/opt/EmbedSky/Qte/x86-qtopia/qtopia/bin/” after compiling is complete.

6. 2. 3 Making desktop starter file

Create a new text file, and add the following contents (the following contents include: program name, icon name and so on). Change the file name into “xxxx.desktop”, and save the file to the directory “$QPEDIR/apps/Applications/”.

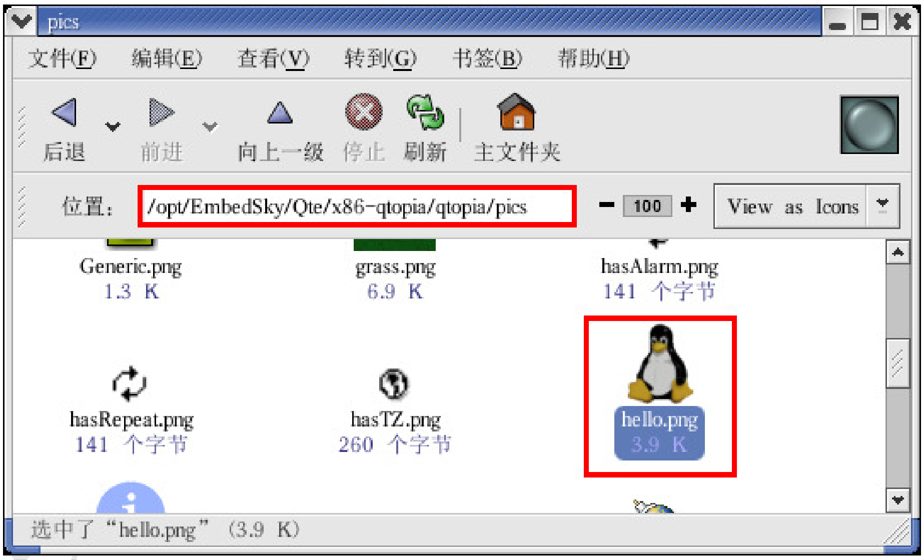
Take the application “hello” for example:

```
[Desktop Entry]
    Comment=A Hello Program
    Exec=hello
    Icon=hello
    Type=Application
    Name=Hello2440
```

6. 2. 4 Making the icon

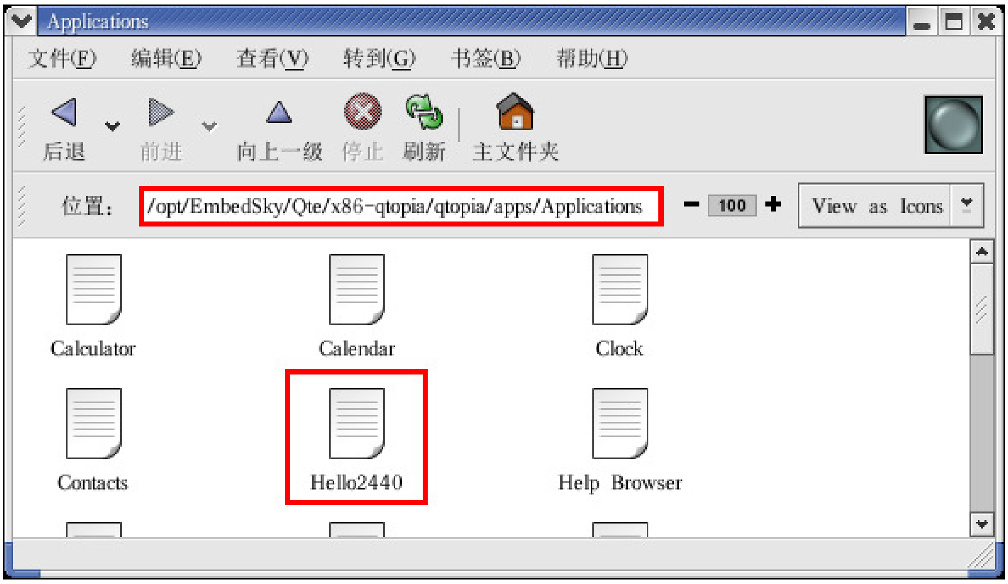
We make a 32×32 icon of PNG format. The naming method is the same as the one naming “xxxx.desktop” introduced in the upper contents. We here name the icon “hello.png” and copy it to the directory

"/opt/EmbedSky/Qte/x86-qtopia/qtopia/pics/":



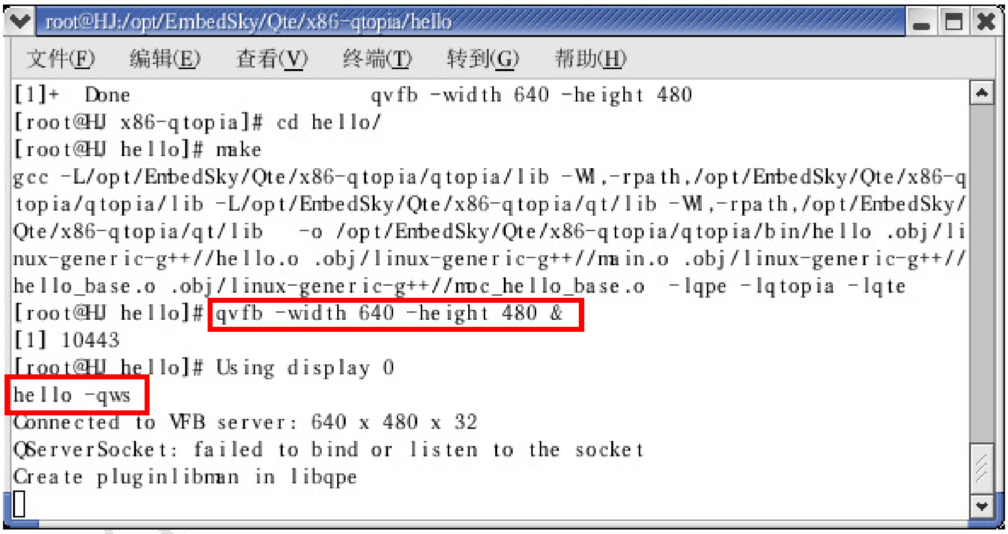
6. 2. 5 Copying Hello.desktop

Copy the finished file "hello.desktop" to "/opt/EmbedSky/Qte/x86-qtopia/qtopia/apps/Applications/".
Copy the desktop starter "Hello2440" to "/opt/EmbedSky/Qte/x86-qtopia/qtopia/ apps/Applications/":

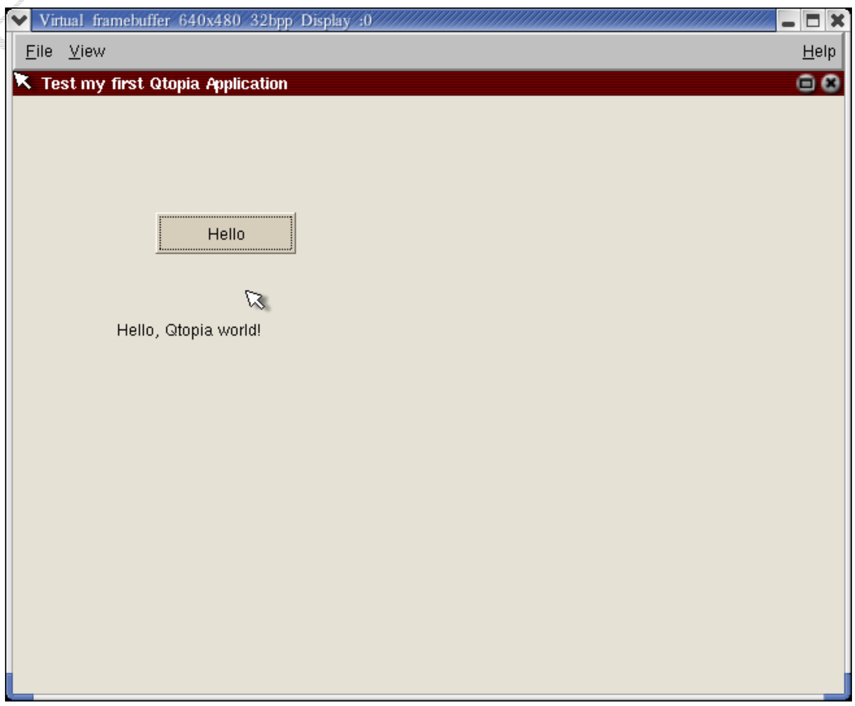


6. 2. 6 Running hello solely

#qvfb –width 640 –height 480 &
#hello –qws

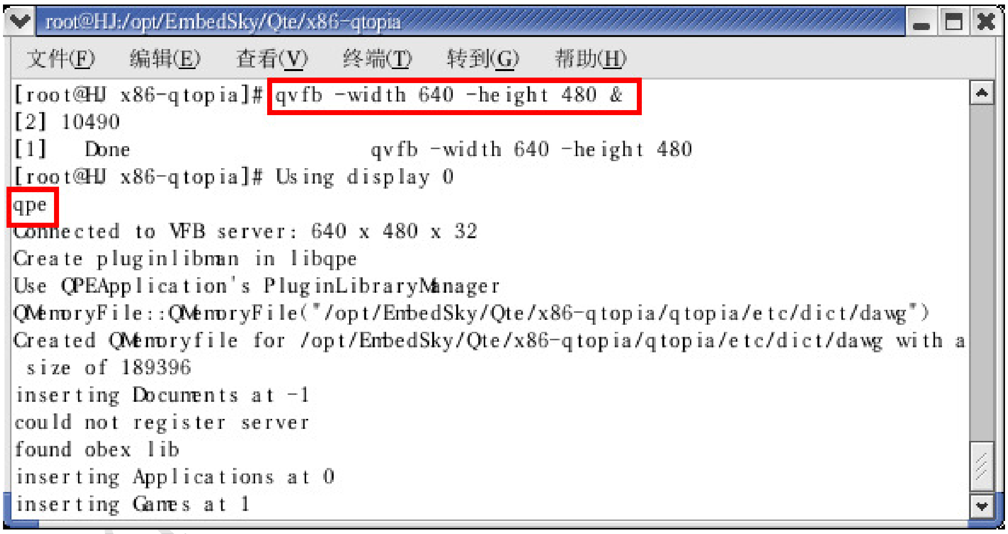


The running interface:

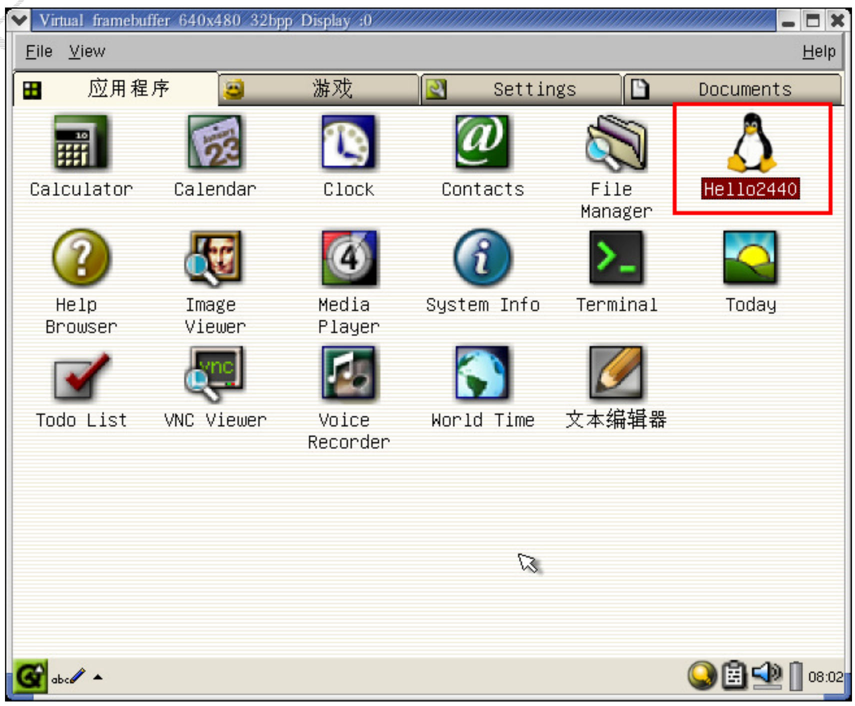


6. 2. 7 Running hello in Qtopia

#qvfb –width 640 –height 480 &
#qpe



The running interface:

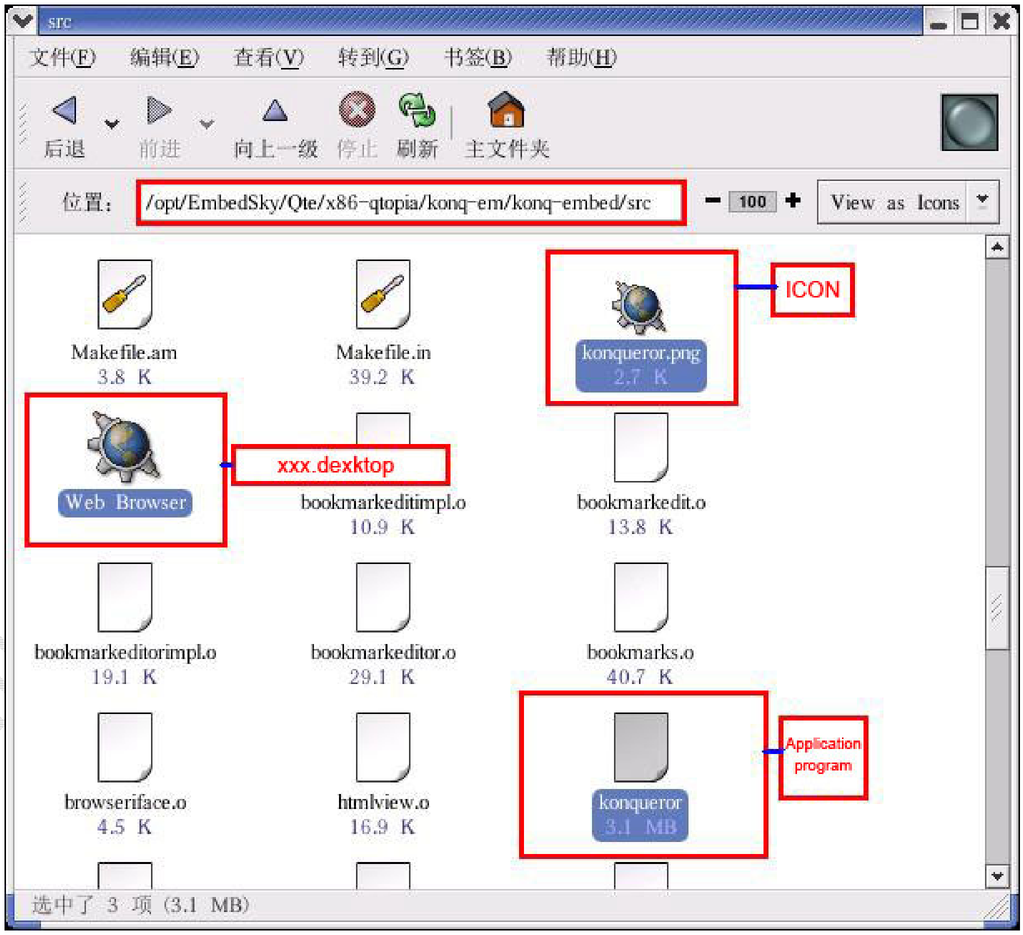


6. 3 Transplanting Web explorer

The Web explorer has been compiled when compiling Qtopia. It is under the directory “konq-em”.

The following contents introduce how to add Web explorer to Qtopia:

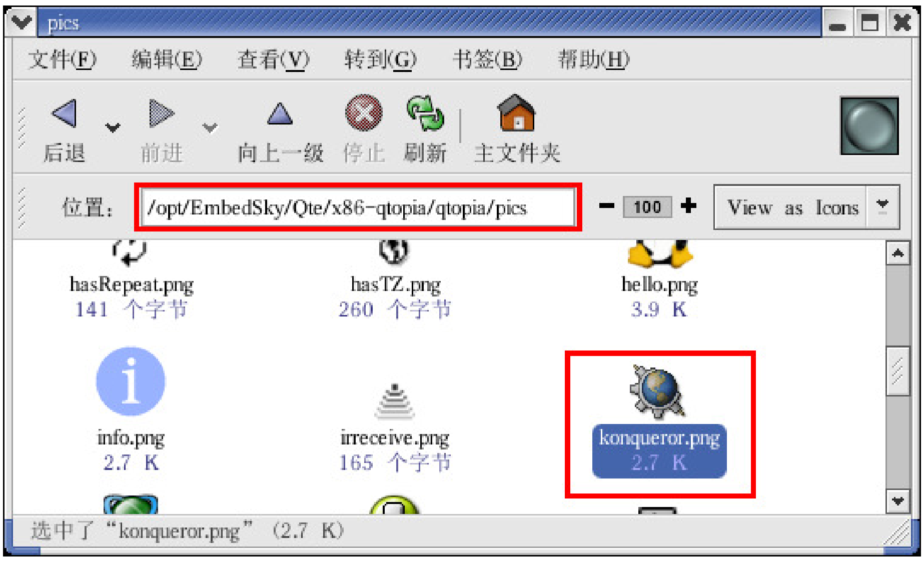
Step1, find following 3 files under the directory “konq-embed/src” of “/opt/EmbedSky/Qte/x86-qtopia/konq-em”:



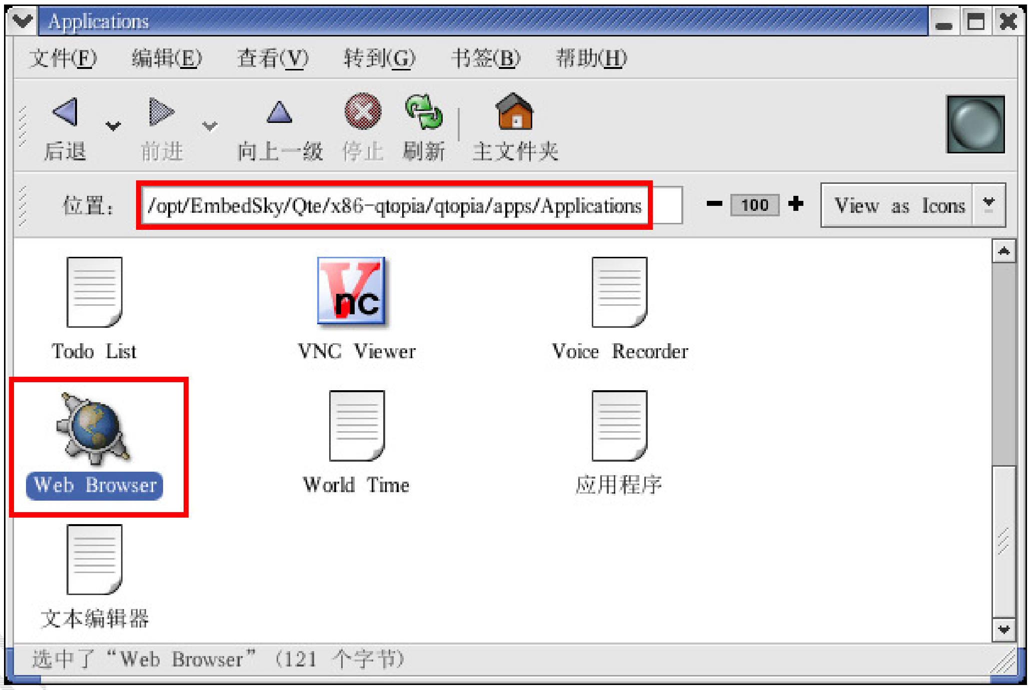
File instruction:

- “konqueror.png” is the icon file of Web explorer.
- “Web Browser” is the desktop starter of Web explorer.
- “conqueror” is the application of Web explorer.

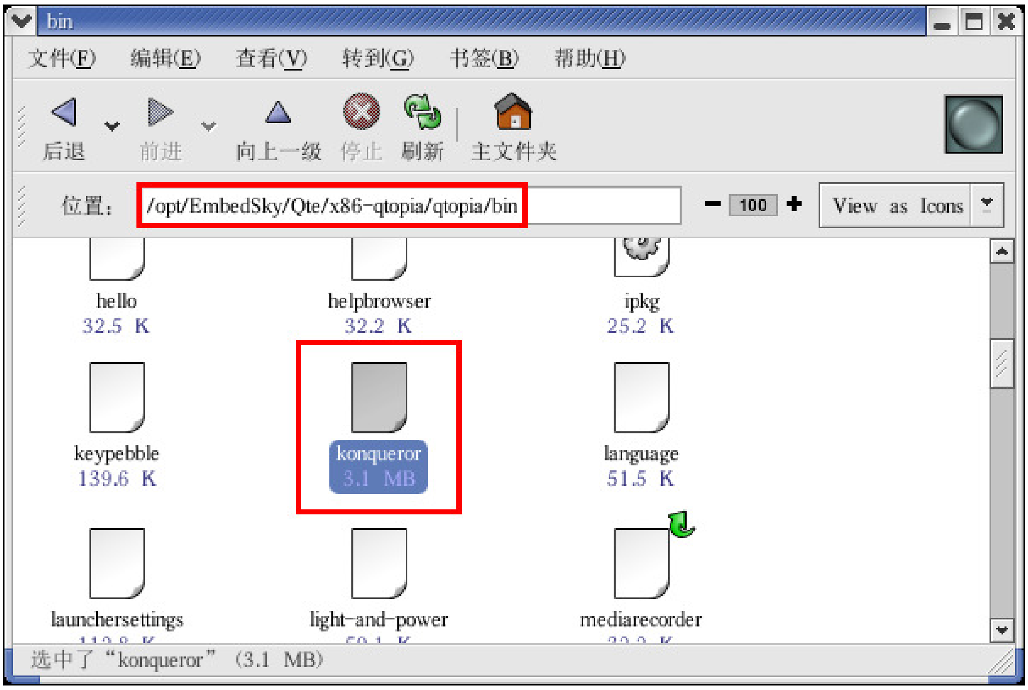
Step2, copy the file “konqueror.png” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/pics/”:



Step3, copy the file “Web Browser” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/apps/Applications/”



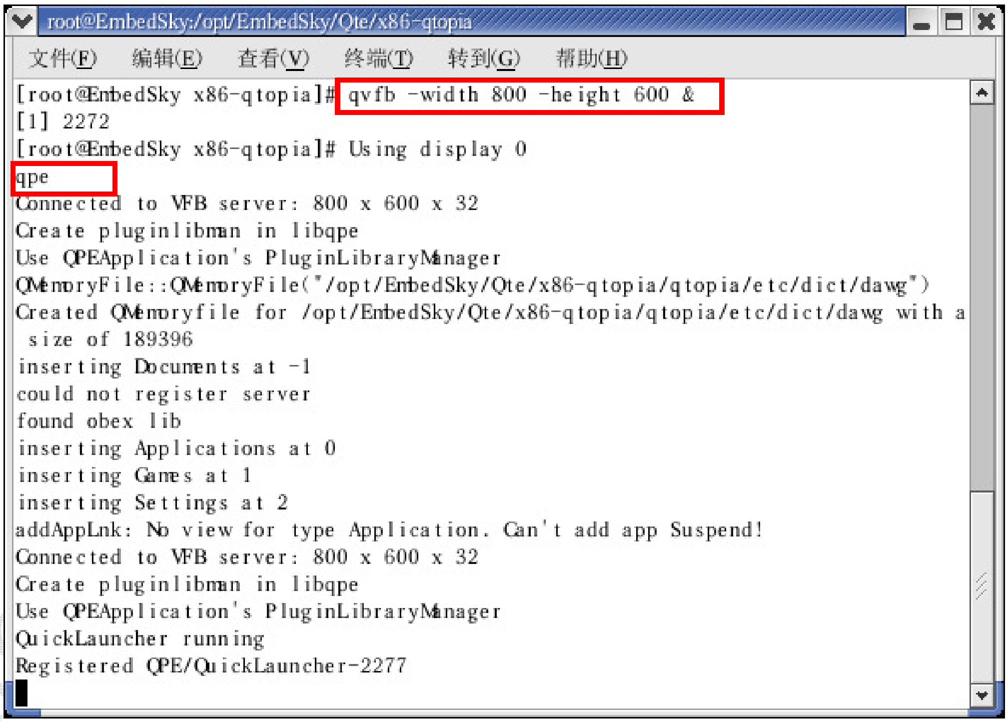
Step4, copy the file “conqueror” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/bin/”:



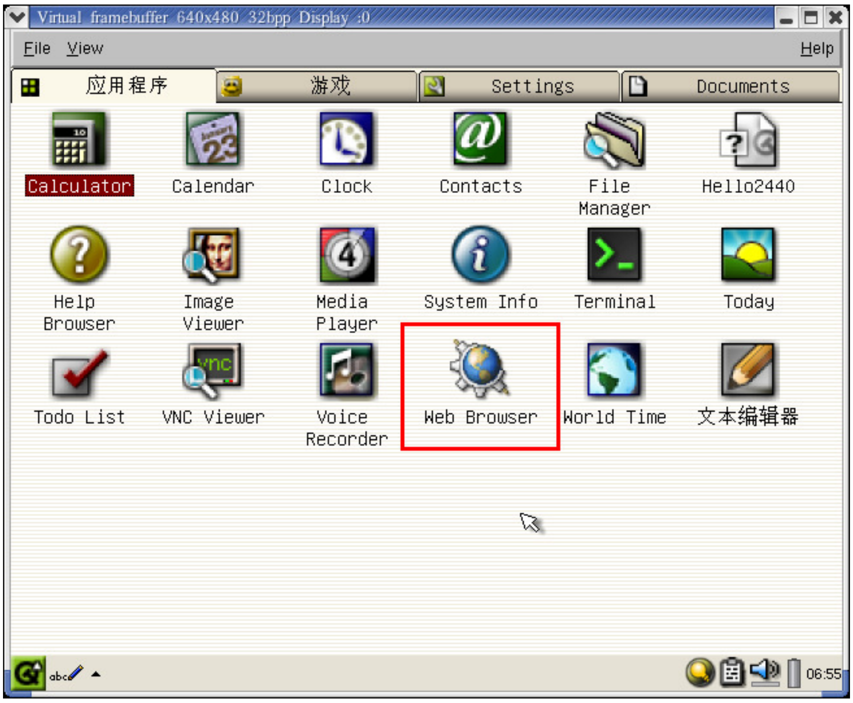
Step5, executing:

#qvfb –width 800 –height 600 &

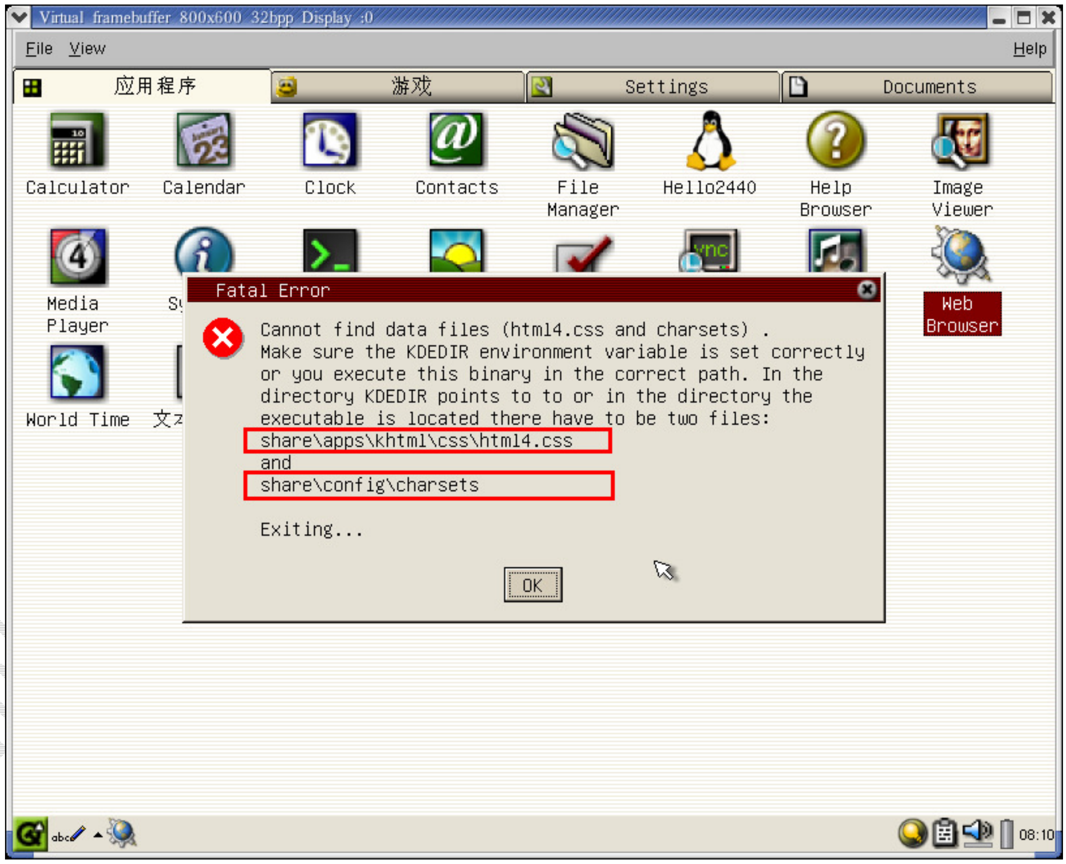
#qpe



Simulating qvfb:

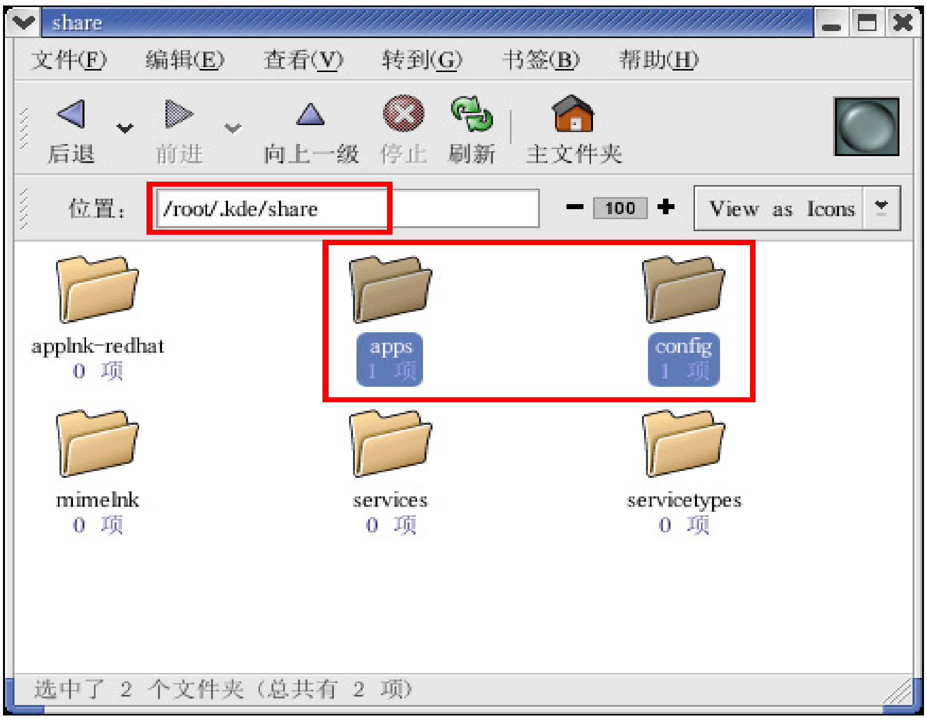


The following errors appear in the first opening of Web explorer



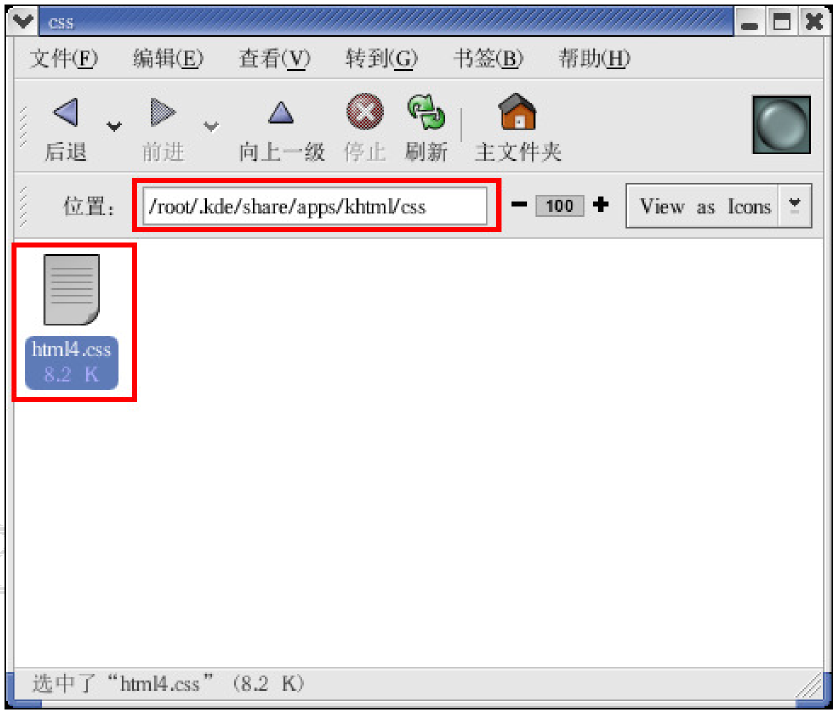
Solution:

1, create 2 directories “apps” and “config” under “/root/.kde/share/”:

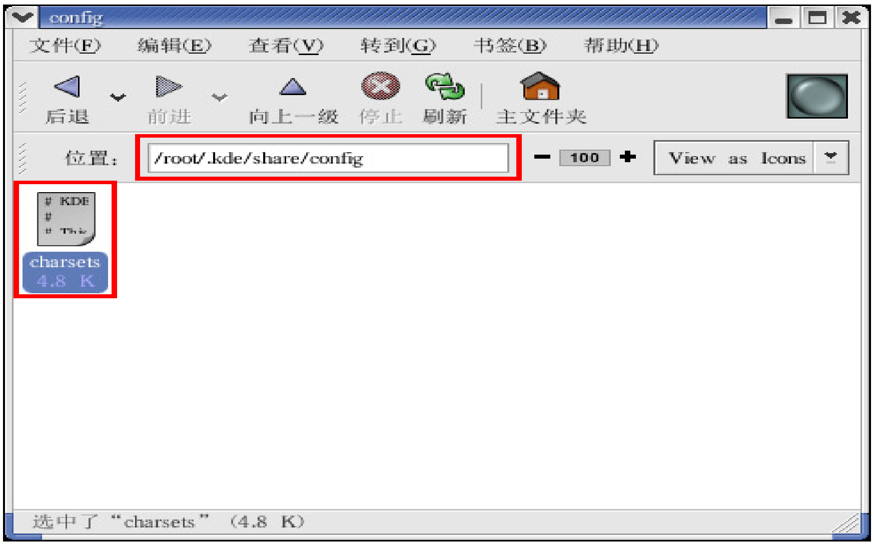


2, create the directory “khtml” under “apps” and create the directory “css” under the directory “khtml”. Copy the file “html4.css” under “/opt/EmbedSky/Qte/x86-qtopia/konq-em/konq-embed/kdesrc/khtml/css/” to

"/root/.kde/share/apps/khtml/css":

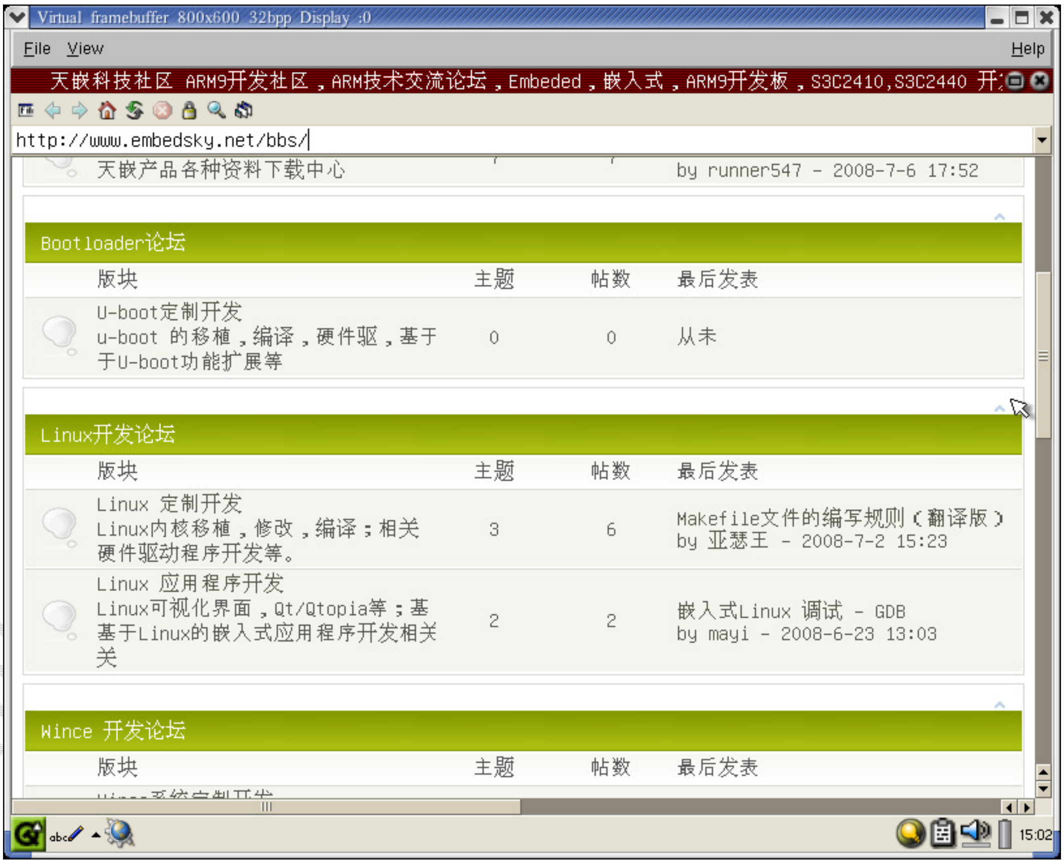


3, copy the file "charsets" under "/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/kdesrc/kdecore/" to "/root/.kde/share/config/":



4, the Web explorer can be accessed successfully after the upper operation.

Access "www.embedsky.net" by using Web explorer:



6. 4 Building development environment based on ARM

We take touch screen version for example here, the same as operations of keyboard and mouse version.

Make sure the cross-compiler version is "3.3.2". The user could enter "arm-linux-gcc –v" in hyper terminal to check the version. If it is not right, please consult "2.5.1" to install the cross-compiler correctly.

6. 4. 1 Installing development environment

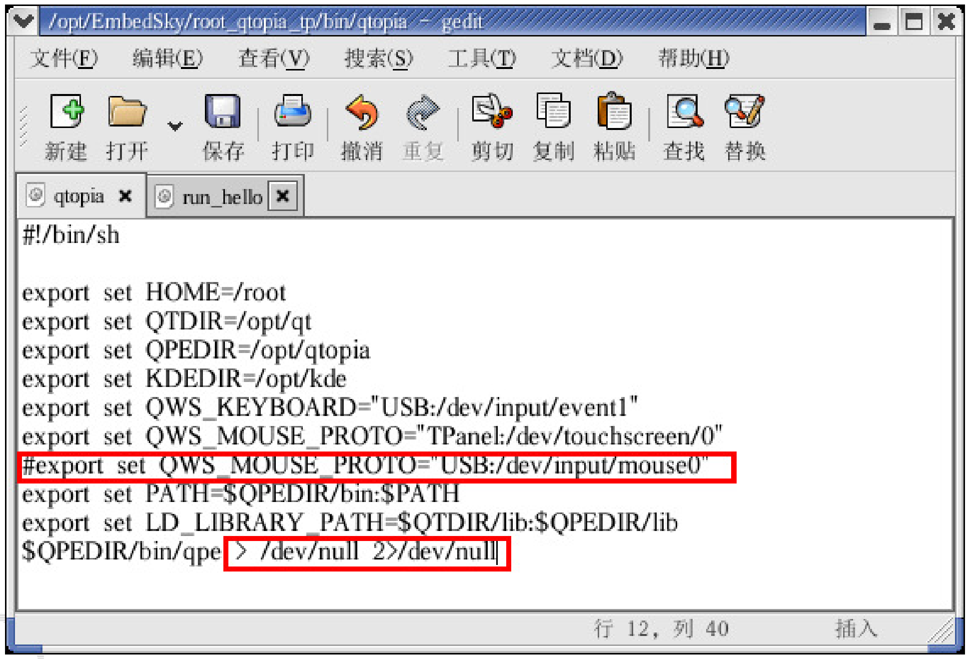
#cd /opt/EmbedSky/Qte/touch-qtopia/

#./build

6. 4. 2 Make running script

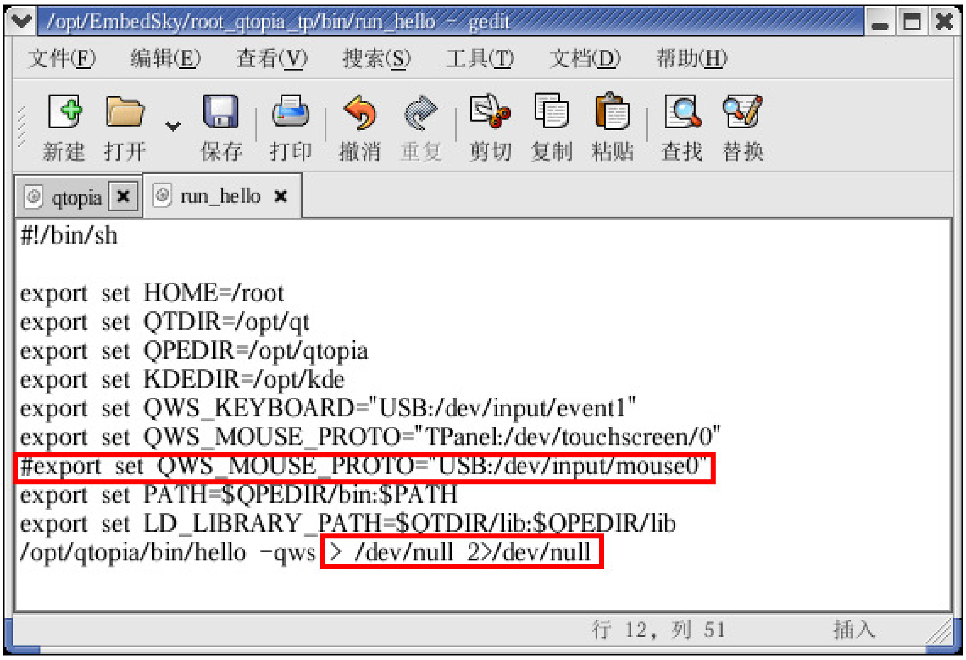
Make the running script to fit Qtopia to the platform.

Qtopia running script:



The command lines “#export set QWS_MOUSE_PROTO=“USB:/dev/input/mouse0”” and “#export set QWS_MOUSE_PROTO=“TPanel:/dev/touchscreen/0”” are corresponding to selecting USB mouse or selecting touch screen. If the user needs qtopis start-up information to be printed, execute the command “> /dev/null 2>/dev/null”.

run_hello running script: (it is used when running “hello” solely)



6. 4. 3 Compiling hello for ARM

cd /opt/EmbedSky/Qte/touch-qtopia/

#. set-env

cd hello

#make

The executable file “hello” is created under the directory “/opt/EmbedSky/Qte/touch-qtopia/qtopia/bin/” after compiling.

6. 4. 4 Installing hello

Copy the executable file “hello” and environment configuration parameters and running script of “hello” to U disk:

#mount /dev/sda1 /mnt/usb //This USB directory is for mounting U disk.

#cp /opt/EmbedSky/Qte/touch-qtopia/qtopia/bin/hello /mnt/usb //Copy the executable file “hello”

#cp /opt/EmbedSky/Qte/touch-qtopia/hello/run_hello /mnt/usb //Copy the running script of “hello”

#cp /opt/EmbedSky/Qte/touch-qtopia/hello/hello.desktp /mnt/usb //Copy the icon file of “hello”

#umount /mnt/usb

Copy “hello” to platform: (insert the U disk containing files of “hello” to USB Host interface on platform)

#mount /dev/sda1 /mnt

#cp /mnt/hello /opt/qtopia/bin //Copy “hello” to “/opt/qtopia/bin ”

#cp /mnt/run_hello /bin //Copy the running script to “/bin”

#cp /mnt/hello.desktop /opt/qtopia/apps/Applications //Copy the icon file to “/opt/qtopia/apps/Applications”

umount /mnt

6. 4. 5 Running hello solely on the platform

Run the running script copied in “6.4.4” and configure the environment parameters. The executable file “hello” runs automatically:

#run_hello &

As shown in the following diagram:

6. 4. 6 Running hello in Qtopia on the platform

Find the qtopia script under the directory “/bin/” of platform and execute it. And then “hello” starts to run.
#qtopia

6. 4. 7 Make desktop starter file

Create a text file and add the following contents: (the following contents include: program name, icon name and so on). Change the file name into “xxxx.desktop”, and save the file to the directory “$QPEDIR/apps/Applications/”.

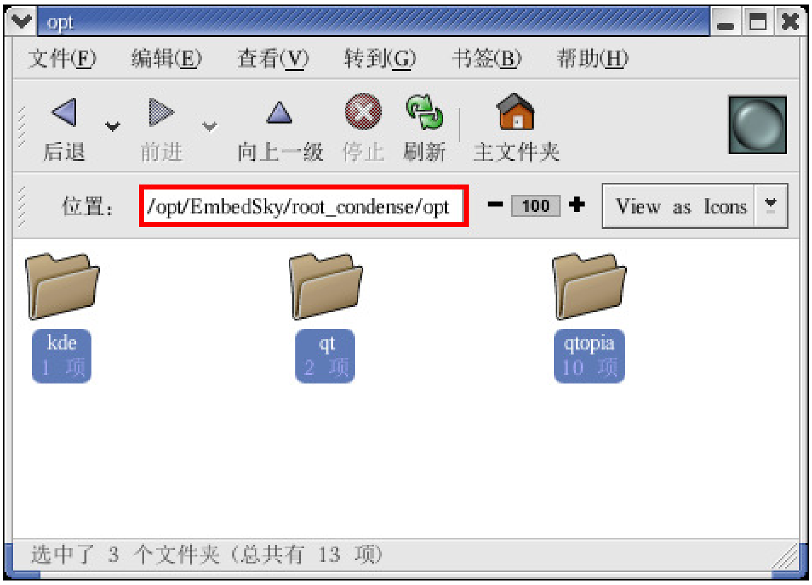
Take the application “hello” for example:

```
[Desktop Entry]
    Comment=A Hello Program
    Exec=hello
    Icon=lyt_demo
    Type=Application
    Name=Hello2440
```

6. 4. 8 Make file system containing Qtopia

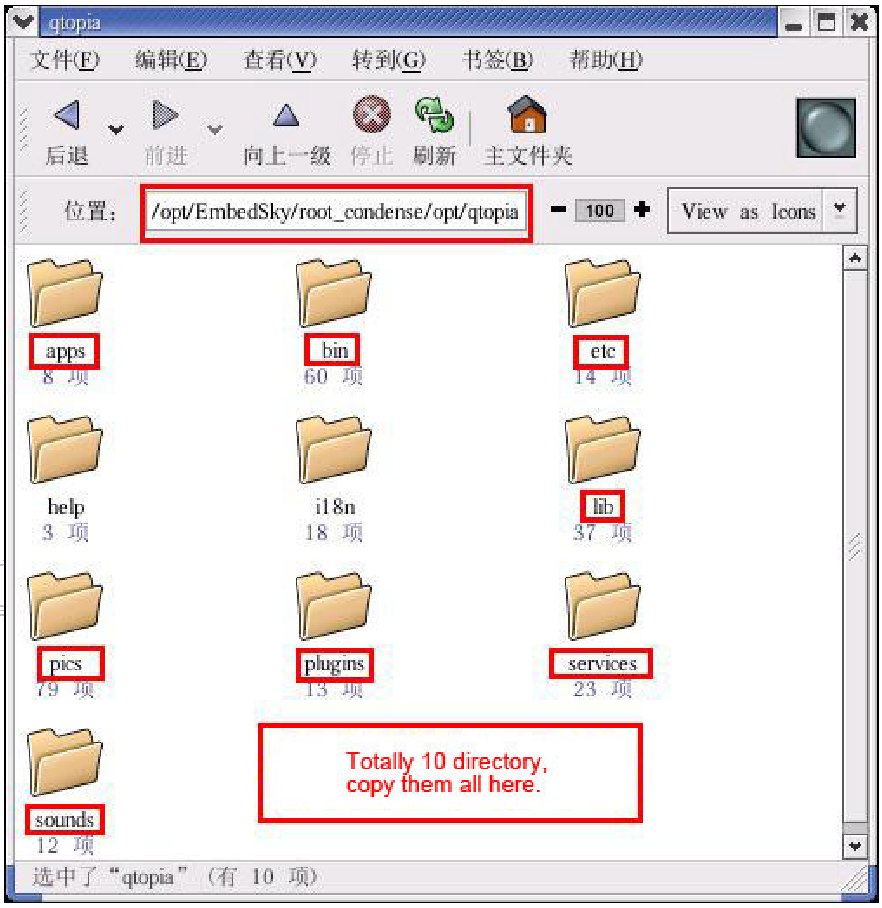
“root_condense” is a simpilied file system. Add the relevant files of Qtopia to “root_condense”, and then we can get the file system containing Qtopia.

Add 3 directories “kde”, “qt” and “qtopia” under the directory “root_condense/opt/”



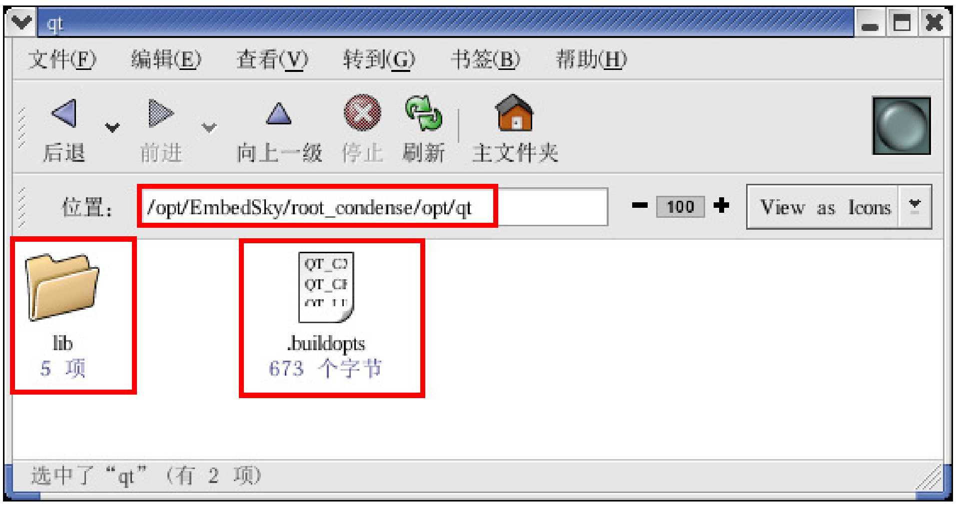
Add Qtopia main application:

Copy directories shown in the following diagram under “/opt/EmbedSky/Qte/touch-qtopia/qtopia/” to “/opt/EmbedSky/root_condense/opt/qtopia/”:

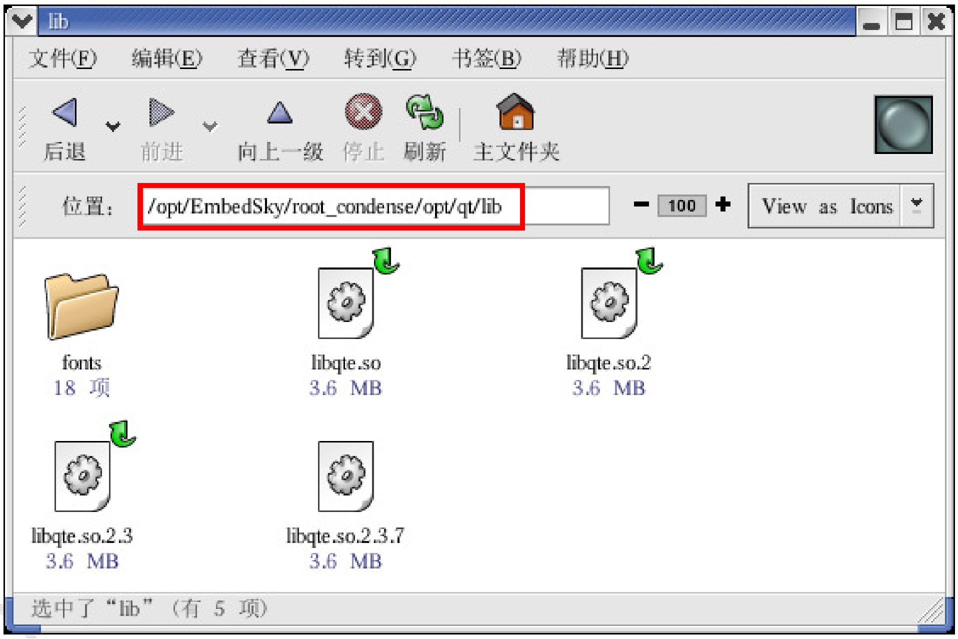


Add libraries needed by Qtopia:

Create the directory “lib” under “/opt/EmbedSky/root_condense/opt/qt/” and copy the file “.buildopts” to “/opt/EmbedSky/root_condense/opt/qt/”:

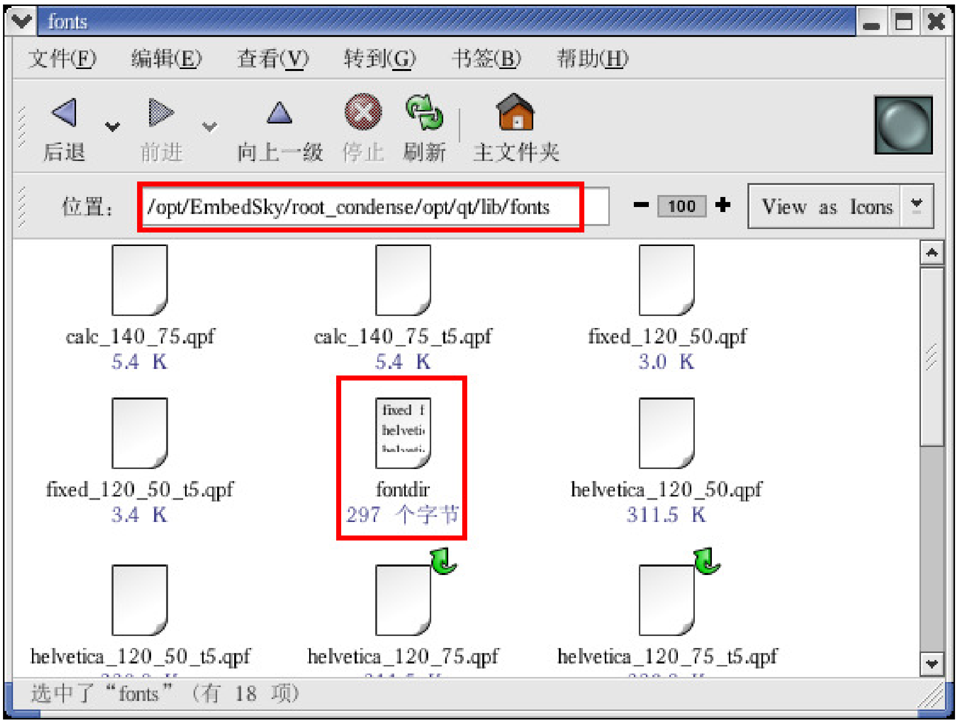


Copy all the files and directories under “/opt/EmbedSky/Qte/touch-qtopia/qt/lib/” to “/opt/EmbedSky/root_condense/opt/qt/lib/”:



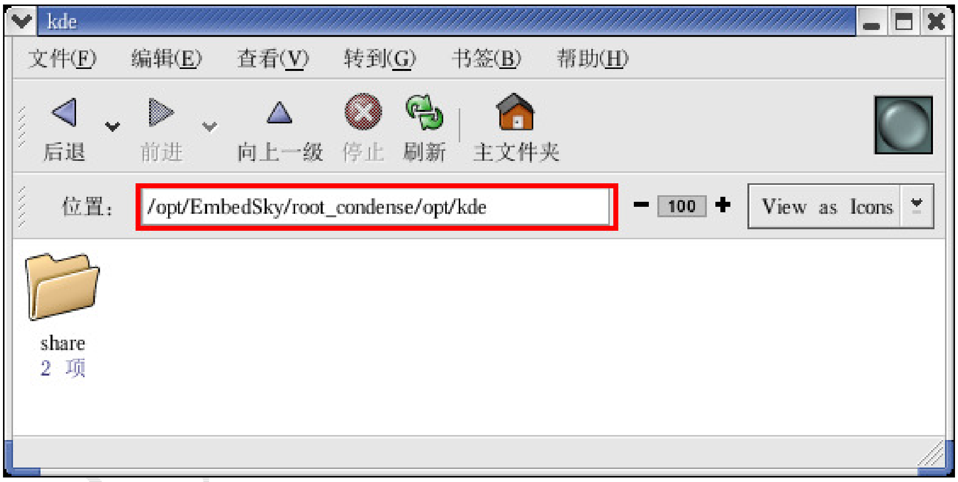
Delete some fonts under “fonts” and modify the file “fontdir” under “file” corresponding with the remaining fonts.

(it is suggested to use the fonts contained in CD-ROM provided by us.)

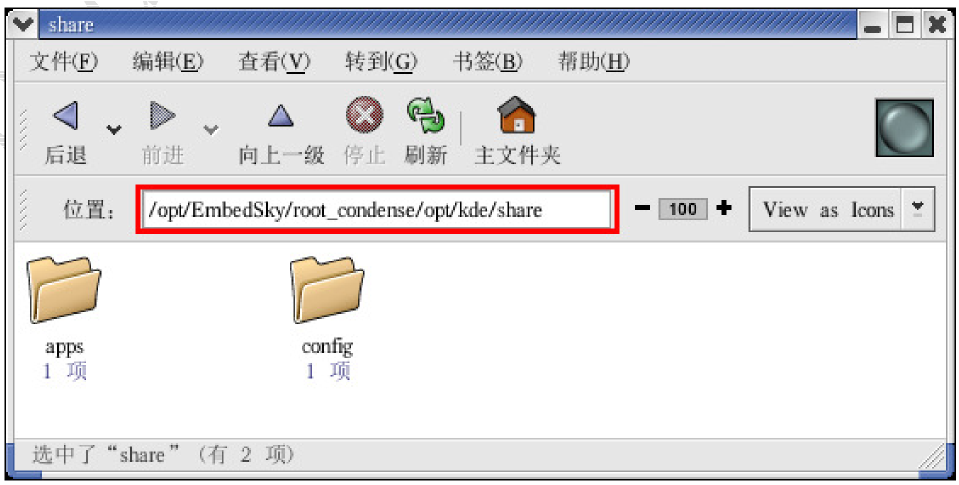


Add Web explorer:

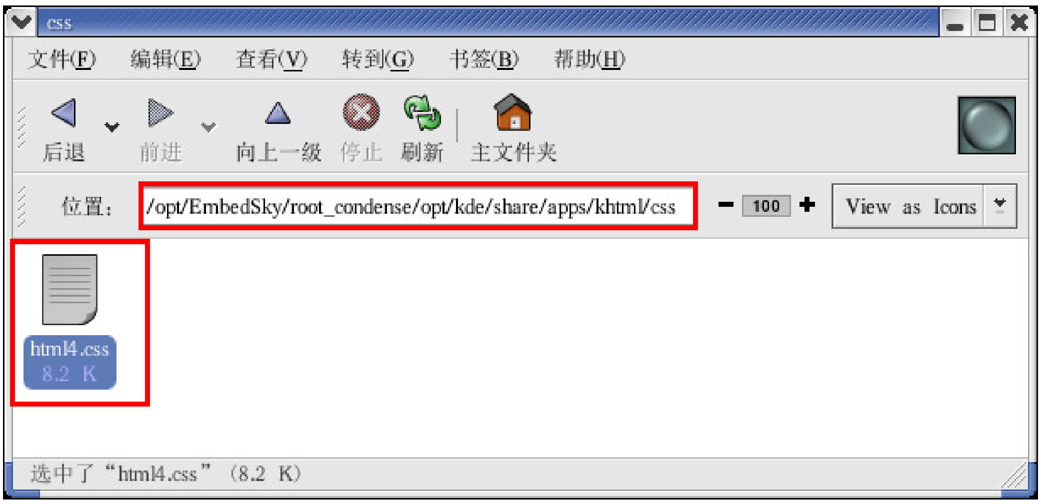
Add the directory “share” under “kde”:



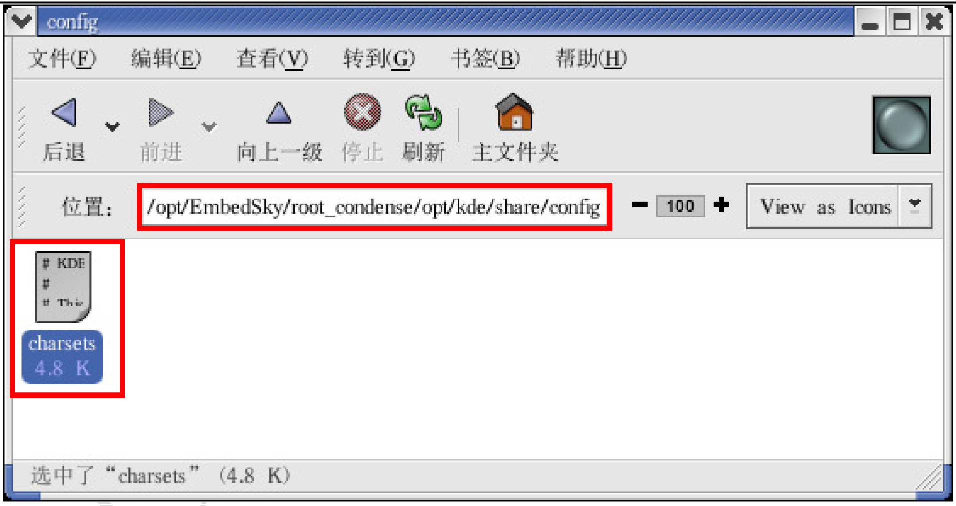
Add 2 directories “apps” and “config” under “share”:



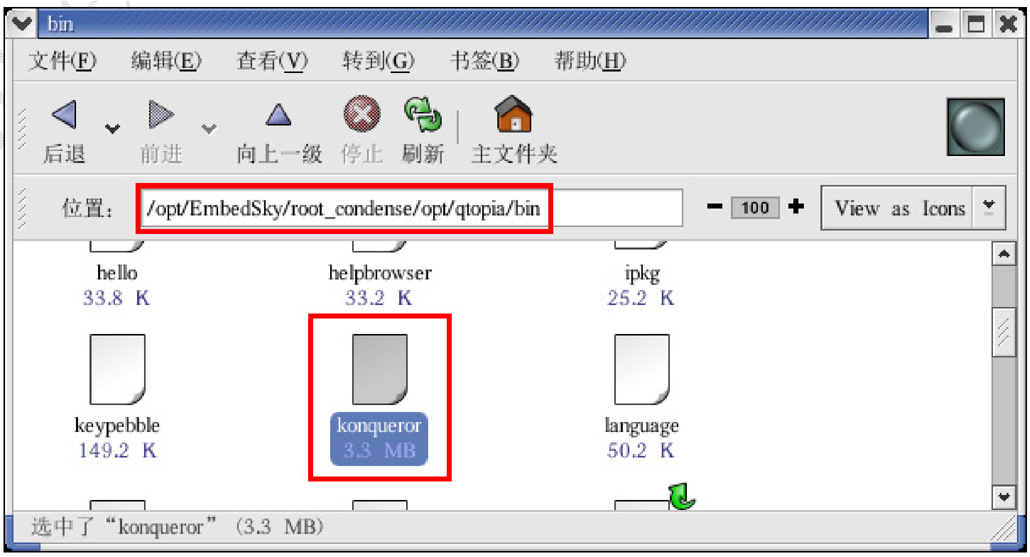
Create “khtml” directory under “apps” and create “css” directory under “khtml”. Copy the file “html4.css” under “/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/kdesrc/khtml/css/” to “/opt/EmbedSky/root_condense/opt/kde/share/apps/khtml/css/”



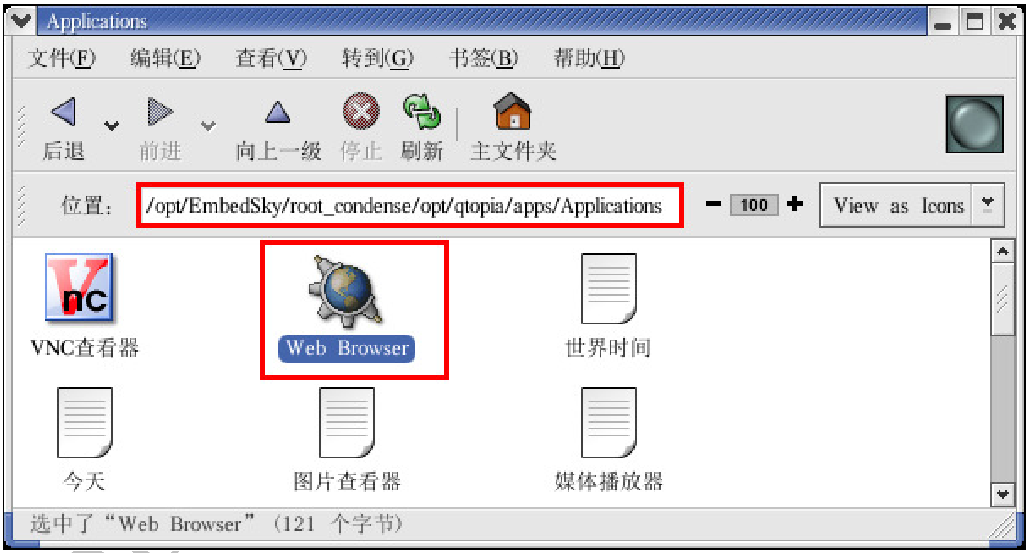
Copy “charsets” under “/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/kdesrc/kdecore/” to “/opt/EmbedSky/root_condense/opt/kde/share/config”:



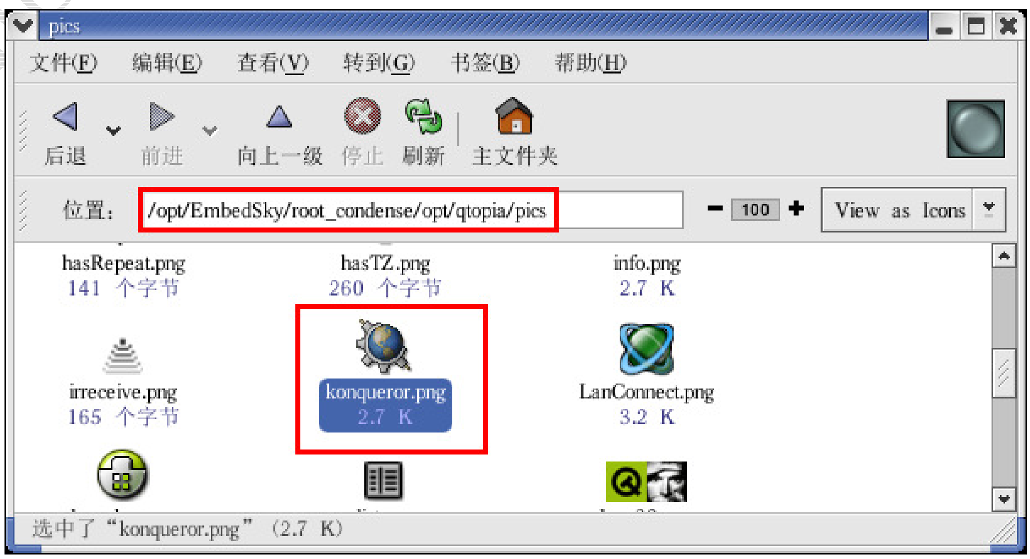
Copy “konqueror” under “/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/src” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/bin/”:



Copy “Web Browser” under “/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/src” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/apps/Applications/”:

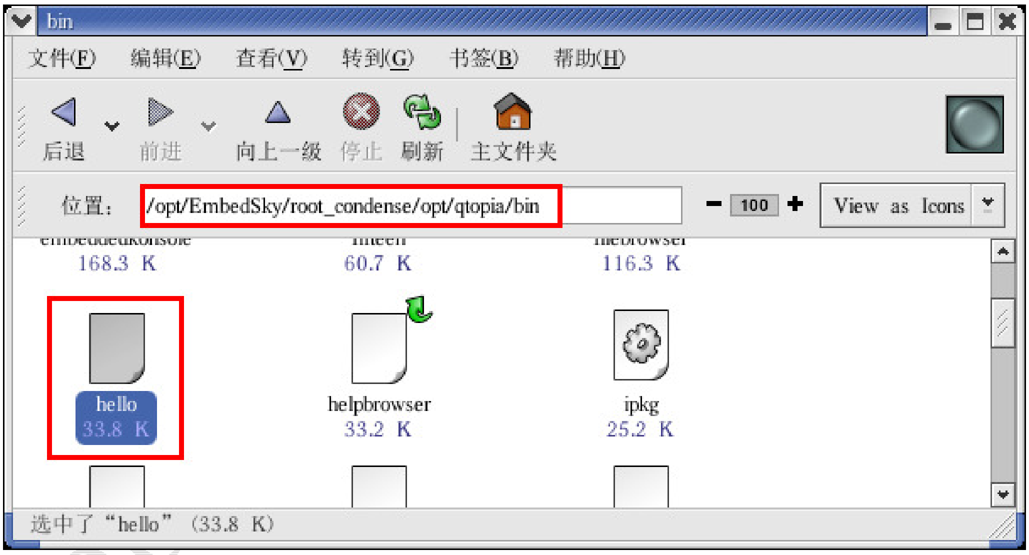


Copy “konqueror.png” under “/opt/EmbedSky/Qte/touch-qtopia/konq-em/konq-embed/src” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/pics/”:

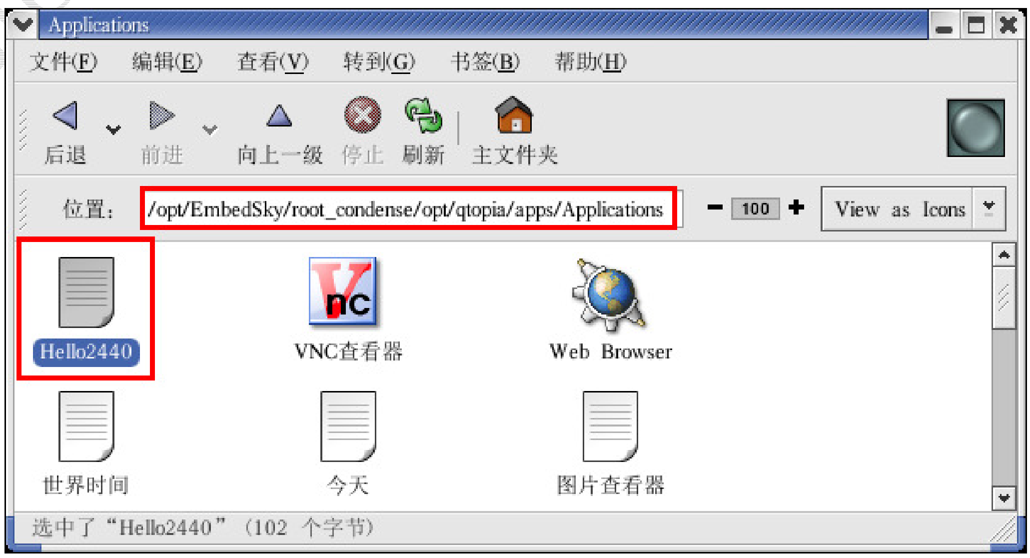


Add the application “Hello”:

The executable file “hello” has been copied to “qtopia/bin/” when compiling “Hello” application:



Copy the desktop starter of hello from “/opt/EmbedSky/Qte/touch-qtopia/hello/” to “/opt/EmbedSky/Qte/x86-qtopia/qtopia/apps/Applications/”:



The file system containing Qtopia is now complete. The user could create a directory “Documents” under “/root_condense/root/” and place files like MP3 under this created directory. After platform start-up, the corresponding files could be found under the menu “Documents” in Qt interface.

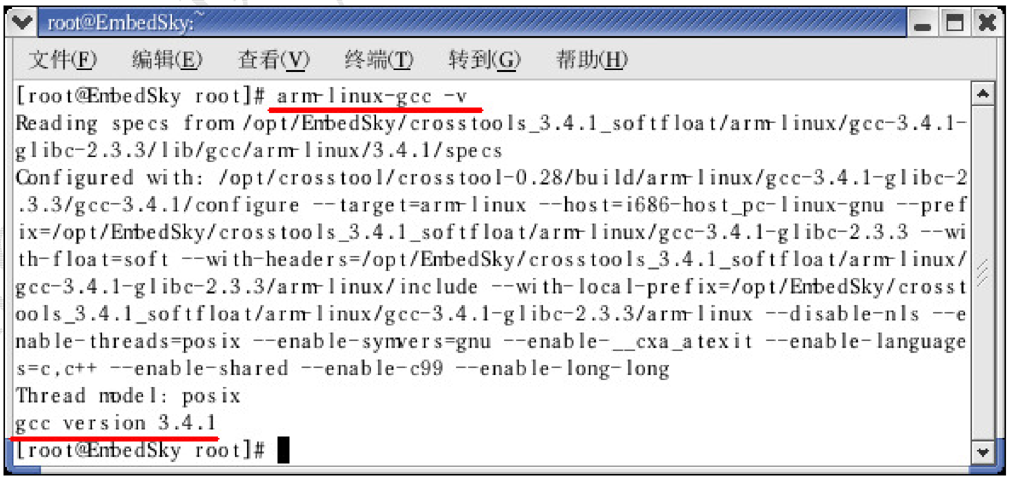
The user can use the software "mkyaffsimage" to make Yaffs file system by following the method introduced previously, and then burn it to the platform for use.

Caution: The touch screen correction is needed when using touch screen file system containing Qt for the first time. The 5 correction points are respectively at upper left, lower left, upper right, lower right and center of the LCD. After touch screen correction, select simplified Chinese as the supporting language.

Chapter 7 Experiment of driver development

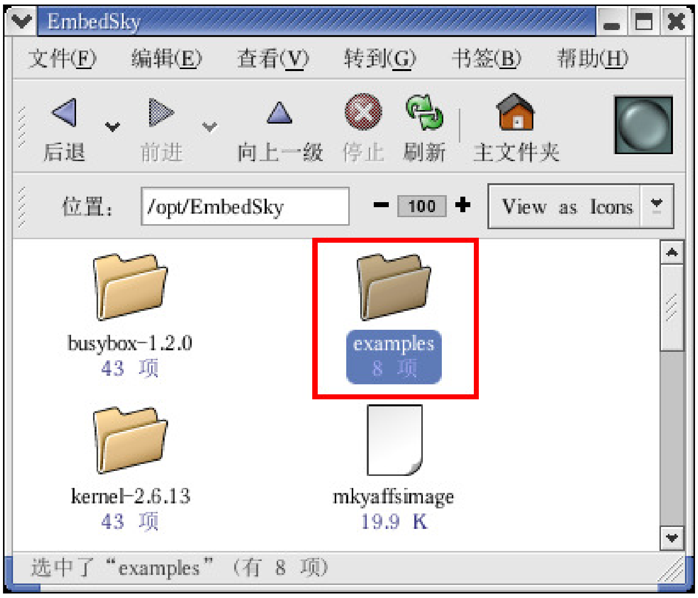
7. 1 Application development in Linux

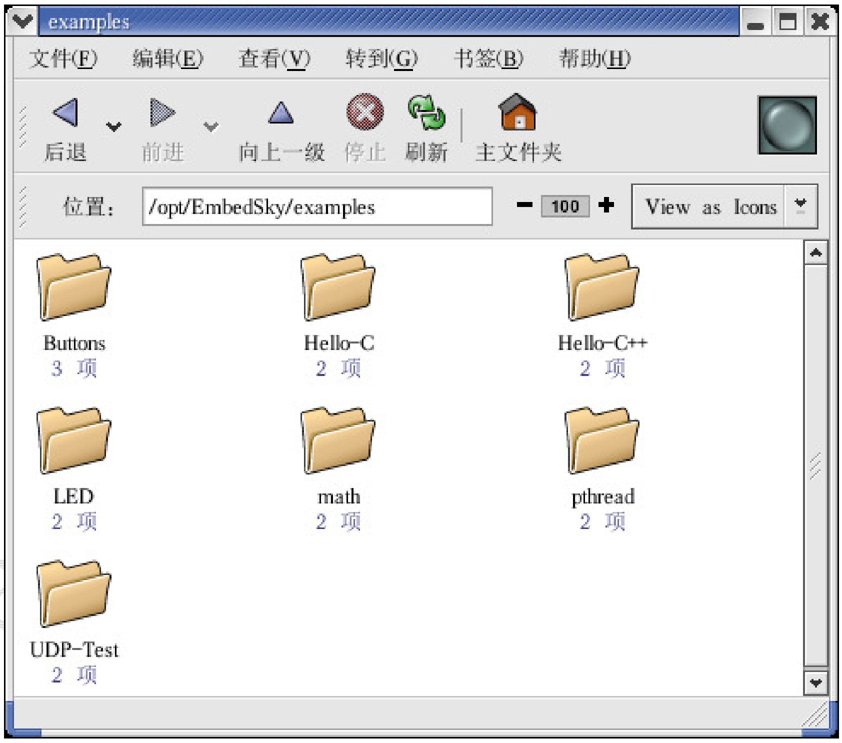
Caution: The cross-compiler of version 3.4.1 is needed for application development under Linux. Make sure you have installed the right version of compiler in PC:



Execute the command "# arm-linux-gcc -v" to check the version of cross-compiler. If the version is not correct, please consult "2.5.1" to re-install the cross-compiler.

The codes mentioned in the following contents can be found in "examples.tar.bz2" of "Linux" in CD-ROM. Locate these files in "/opt/EmbedSky/examples/" after decompressing "examples.tar.bz2":





7. 1. 1 Hello EmbedSky experiment

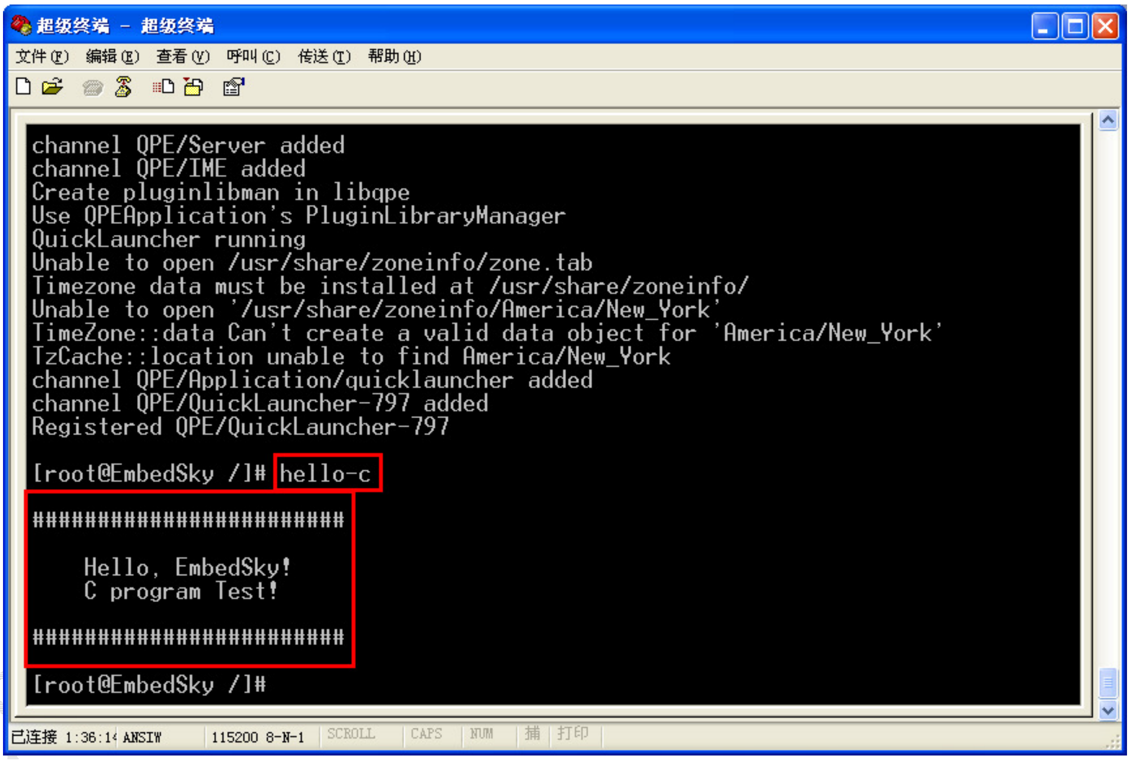
Hello EmbedSky includes experiments of C language and the ones of C++.

➢ Experiment of C language:

The following C source code is under the directory “/opt/EmbedSky/examples/Hello-C/hello-c.c”

```
/*************************************

NAME:hello-c.c
COPYRIGHT:www.embedsky.net

*************************************/

#include <stdio.h>

int main(void)
{
	printf("\n########################\n");			//Print information
	printf("\n	Hello, EmbedSky!\n");
	printf("	C program Test!\n");
	printf("\n########################\n\n");
}
```

Execute the command “arm-linux-gcc –o hello-c hello-c.c” or “make” to cross compile the code. And then run the compiled program “hello-c”:



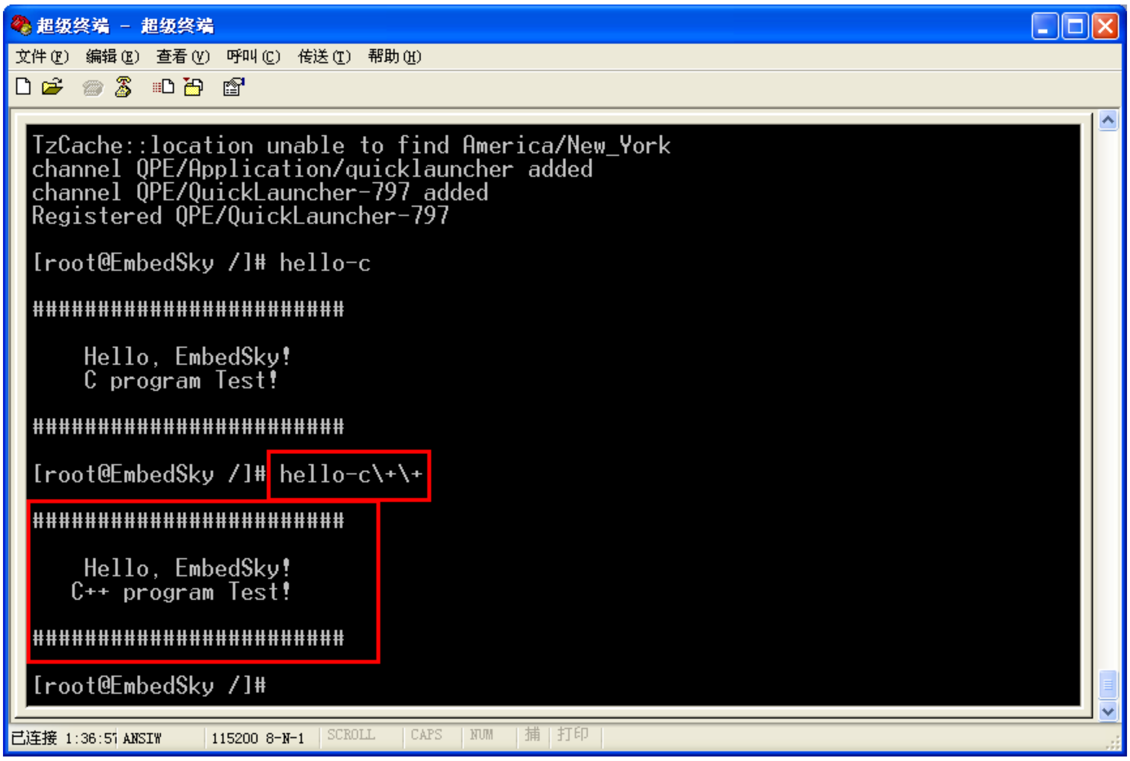
➤ Experiment of C++:

The following C++ source code is under the directory "/opt/EmbedSky/examples/Hello-C++/hello-c++.c++":

```
/***************************************

NAME:hello-c++.c++
COPYRIGHT:www.embedsky.net

***************************************/
#include <iostream>
#include <cstring>
using namespace std;

class String
{
    private:
        char *str;
    public:
        String(char *s)                                    //Input character string
        {
            int lenght=strlen(s);
            str = new char[lenght+1];
            strcpy(str, s);
        }
```

```
        void display()                              //Function of printing information
        {
        cout << str <<endl;
        }
};

int main(void)
{
    String s1 = "\n#########################\n";        //Character string
    String s2 = "       Hello, EmbedSky!";
    String s3 = "      C++ program Test!";
    String s4 = "\n#########################\n";

    s1.display();//Call the function of printing
    s2.display();
    s3.display();
    s4.display();
    return 0;
}
```

Execute the command “arm-linux-g++ –o hello-c++ hello-c++.c++” or “make” to cross compile the code. And then run the compiled program “hello-c”:

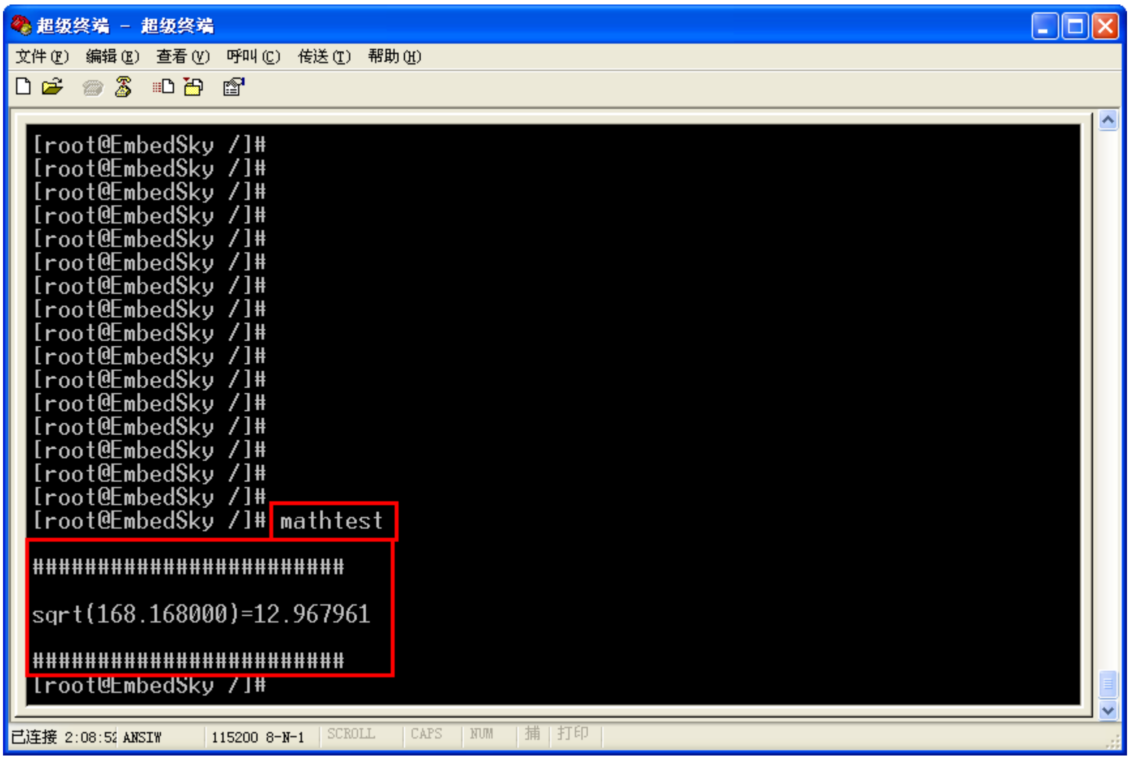


7. 1. 2 Experiment of calling math function

The following test code calls the squaring root function.

```
/**************************************

NAME:mathtest.c
COPYRIGHT:www.embedsky.net

**************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void)
{
	double a=168.168;
	printf("\n########################\n");		//Print information
	printf("\nsqrt(%f)=%f\n", a, sqrt(a));		//Call squaring root function
	printf("\n########################\n\n");
	return 0;
}
```

Execute the command "arm-linux-gcc –o mathtest mathtest.c -lm" (be cautious of the parameter "-lm") or "make" to cross compile the code. And then run the compiled program "mathtest":



7. 1. 3 Experiment of thread programming

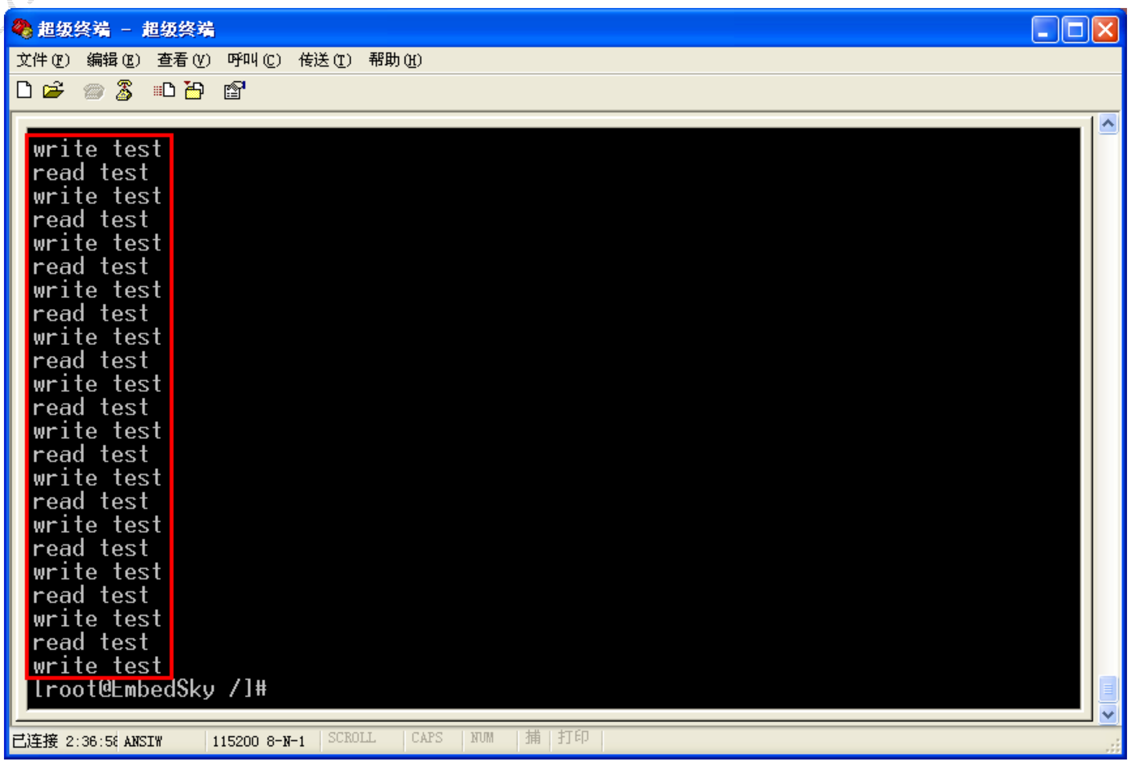
The brief introduction of the following thread programming example: A thread reads data from a shared buffer and prints them out; and another thread writes data into the shared buffer at the same time and prints them out. The shared buffer access is based on mutex principle.

The source code:

```
/*************************************

NAME:pthread.c
COPYRIGHT:www.embedsky.net

*************************************/
#include<stddef.h>
#include<stdio.h>
#include<unistd.h>
#include"pthread.h"

void reader_function(void);
void writer_function(void);
char buffer;
int buffer_has_item=0;
pthread_mutex_t mutex;
main()
{
	pthread_t reader;
	pthread_mutex_init(&mutex,NULL);
	pthread_create(&reader,NULL,(void*)&reader_function,NULL);	//
	writer_function();
}
void writer_function(void)
{
	while(1)
	{
		pthread_mutex_lock(&mutex);
		if(buffer_has_item==0)
		{
		buffer='s';
		printf("write test\n");
		buffer_has_item=1;
		}
	pthread_mutex_unlock(&mutex);
	}
```

```
}
void reader_function(void)
{
        while(1)
        {
                pthread_mutex_lock(&mutex);
        if(buffer_has_item==1)
                {
                buffer='\0';
                printf("read test\n");
                buffer_has_item=0;
                }
                pthread_mutex_unlock(&mutex);
        }
}
```

Execute the command "arm-linux-gcc -static -o pthread pthread.c -lpthread" (caution: the parameter "-lpthread" is indispensable. "-lpthread" means calling the library "libpthread") or "make" to cross compile the code. And then run the compiled program "mathtest":



7. 1. 4 Experiment of UDP network programming

TCP/IP provides a connectionless transport layer protocol: UDP (User Datagram Protocol). The difference between UDP and TCP/IP is caused by the difference between connectionless socket programming and connection-oriented socket programming. Every single receiving or sending UDP package contains the address of sender and receiver.

A data package socket of class “SOCK_DGRAM” needs to be created before transporting, by calling the following function:

sockfd = socket(AF_INET,SOCK_DGRAM,0);

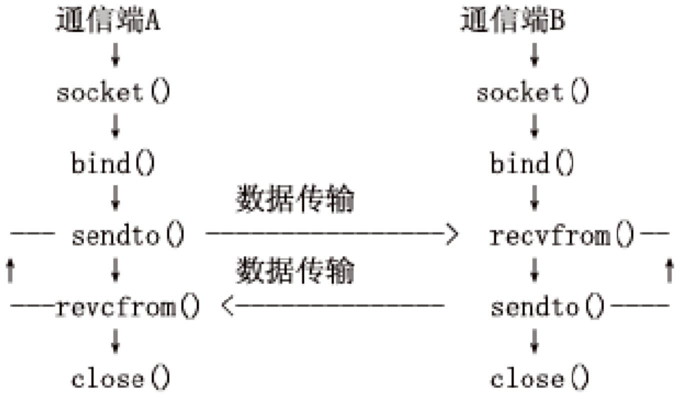
The operation of sending and receiving begins right after the socket creating because of connectionless. The receiver needs to tell sender the receiving port. The 2 functions “sendto” and “recvfrom” are used for sending and receiving. The following contents show how to call these 2 functions:

int sendto (int s, const void *msg, int len, unsigned int flags, const truct sockaddr *to, int tolen);

int recvfrom (int s,void *buf, int len, unsigned int flags, struct sockaddr *from, int formlen);

“s” is the name of socket; “msg” and “buf” point to sending buffer and receiving buffer respectively; “len” represents the length of buffer; “flag” is the option flag. The value is 0 here because it is not used in the example; “to” and “from” point to the destination and source address, including IP address and port information; “tolen” and “fromlen” represent the length of sending and receiving socket address structure. The 2 functions return the length of sending or receiving bytes. The return value is -1 when error occurs.

The general process of connectionless transmission:



In upper diagram, the sender and receiver have bound their address ports. However, in some conditions, one side of the sender or receiver does not have to bind the address port which can be allocated by kernel. During the communication, the side without binding sends the data package preceding the other side, and the receiver extracts port information of the sender from the package in order to get to know the exactly address of the un-bingding side.

The same as function read() and write(), process blocking always occurs in recvfrom() and sendto(). And it is possible to receive an empty package, which is different from TCP/IP. In this case, the application set “msg” of sendto() into “NULL” and set “len” into “0”.

The UDP example source code:

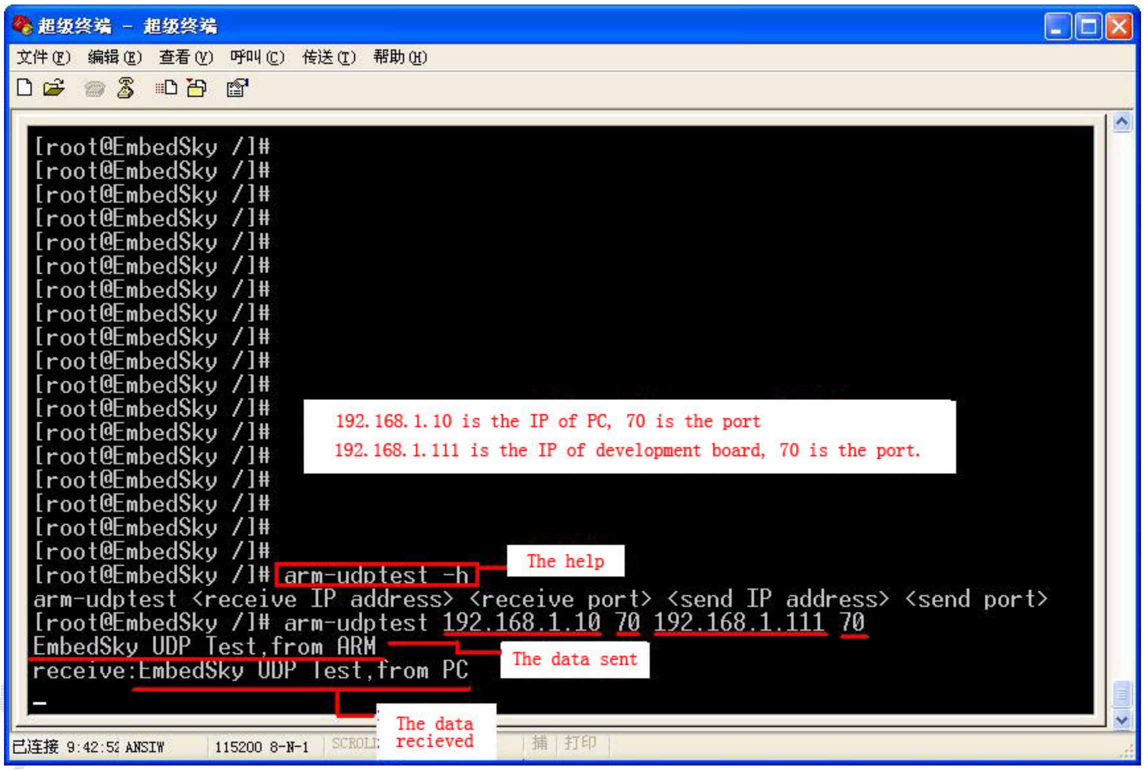
```
/*************************************

NAME:UDP.c
COPYRIGHT:www.embedsky.net

*************************************/
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <arpa/inet.h>
#include <stdio.h>

#define BUFLEN 255

int main(int argc, char **argv)
{
	struct sockaddr_in peeraddr, localaddr;
//peeraddr is used to preserve the IP and port address of the peer; localaddr is used to preserve the local socket address.
	int sockfd;
	char recmsg[BUFLEN+1];
	int socklen, n;

	if(argc!=5){
		printf("%s <receive IP address> <receive port> <send IP address> <send port>\n", argv[0]);
		exit(0);
	}

	sockfd = socket(AF_INET, SOCK_DGRAM, 0);
	if(sockfd<0){
		printf("socket creating err in udptest\n");
		exit(1);
	}
	socklen = sizeof(struct sockaddr_in);
	memset(&peeraddr, 0, socklen);
	peeraddr.sin_family=AF_INET;
	peeraddr.sin_port=htons(atoi(argv[2]));
	if(inet_pton(AF_INET, argv[1], &peeraddr.sin_addr)<=0){
		printf("Wrong receive IP address!\n");
		exit(0);
	}
	memset(&localaddr, 0, socklen);
	localaddr.sin_family=AF_INET;
	if(inet_pton(AF_INET, argv[3], &localaddr.sin_addr)<=0){
		printf("Wrong send IP address!\n");
		exit(0);
	}
	localaddr.sin_port=htons(atoi(argv[4]));
	if(bind(sockfd, &localaddr, socklen)<0){
		printf("bind local address err in udptest!\n");
		exit(2);
	}
```

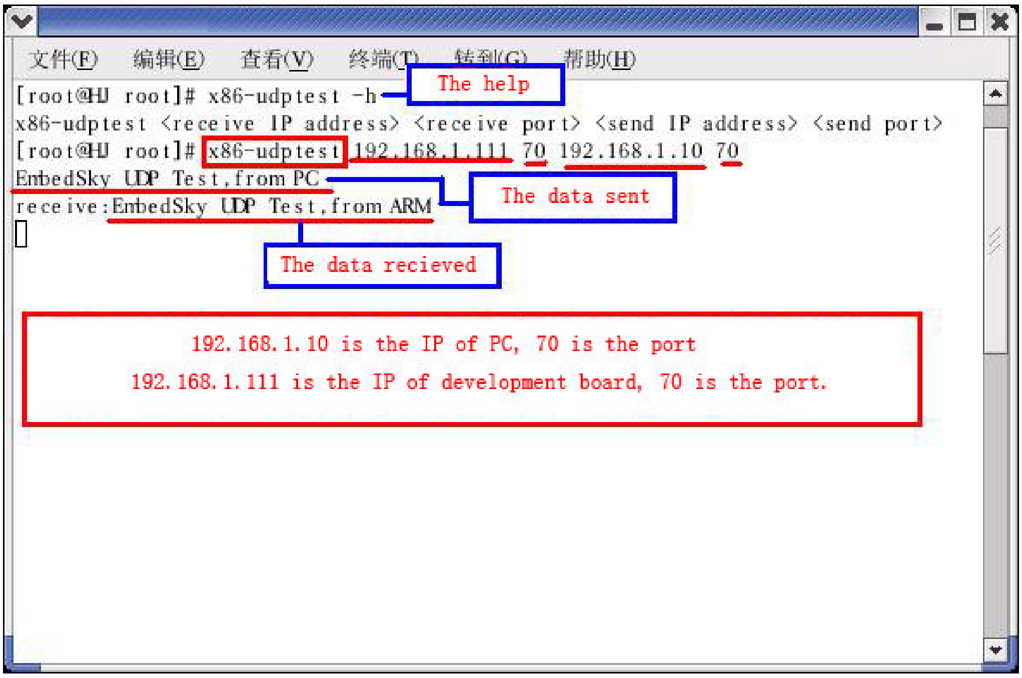
```
	if(fgets(recmsg, BUFLEN, stdin) == NULL) exit(0);
	if(sendto(sockfd, recmsg, strlen(recmsg), 0, &peeraddr, socklen)<0){
		printf("sendto err in udptest!\n");
		exit(3);
	}

	for(;;){
		/*recv&send message loop*/
		n = recvfrom(sockfd, recmsg, BUFLEN, 0, &peeraddr, &socklen);
		if(n<0){
			printf("recvfrom err in udptest!\n");
			exit(4);
		}else{
//receiving package successfully
			recmsg[n]=0;
			printf("receive:%s", recmsg);
		}
		if(fgets(recmsg, BUFLEN, stdin) == NULL) exit(0);
		if(sendto(sockfd, recmsg, strlen(recmsg), 0, &peeraddr, socklen)<0){
			printf("send to err in udptest!\n");
			exit(3);
		}
	}
}
```

Compile the test program running in the platform and PC by executing “arm-linux-gcc -o arm-udptest UDP.c” and “gcc –o x86-udptest UDP.c” or by executing the command “make”.

Run “arm-updtest” in the platform:



Run “x86-udptest” in PC:



7.1.5 Experiment of controlling LED

LED controlling application manages the start-up, writing and shut-down action of LED driver, in order to control LED swiching on and swithing down.

The source code:

```
/*************************************

NAME:leds.c
COPYRIGHT:www.embedsky.net

*************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>

int main(int argc, char **argv)
{
	int on;
	int led_no;
	int fd;
	if (argc != 3 || sscanf(argv[1], "%d", &led_no) != 1 || sscanf(argv[2],"%d", &on) != 1 ||
		on < 0 || on > 1 || led_no < 0 || led_no > 3)
//Check the 2 parameters of LED controlling. Exit if no parameter input.
{
		fprintf(stderr, "Usage: leds led_no 0|1\n");
		exit(1);
	}
	fd = open("/dev/TQ2440_leds", 0);		//open the device file "/dev/TQ2440_leds"
	if (fd < 0) {
		perror("open device leds");
		exit(1);
	}
	ioctl(fd, on, led_no);			//Control LED by calling "ioctl" and inputting parameter
	close(fd);				//shut down the device
	return 0;
}
```

Cross-compile the test program by executing “arm-linux-gcc -o leds leds.c” or “make”, and download the executable program “leds” to platform and execute it.

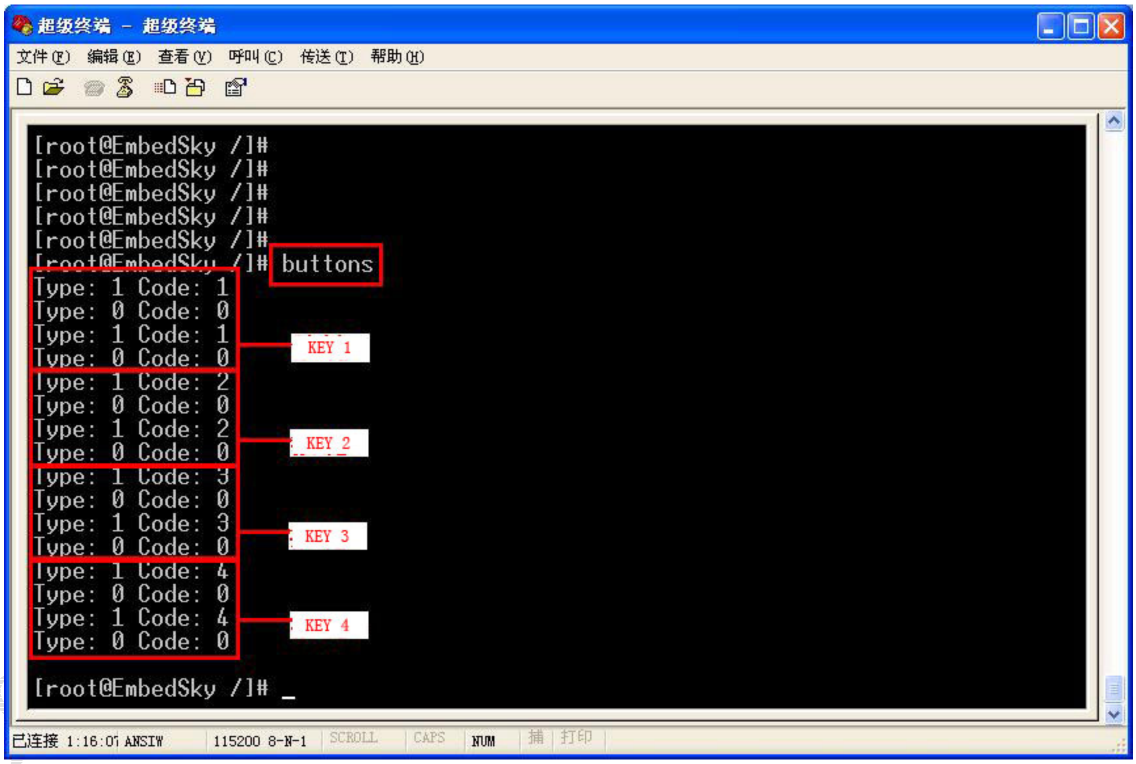
7. 1. 6 Experiment of user button controlling

The button controlling program starts the device by using blocking method. The controlling program is blocked into “read” function when no press-action happens. When press-action is detected, “read” function returns, and the program outputs action code by calling “printf” function.

The source code:

```
/*************************************

NAME:buttons.c
COPYRIGHT:www.embedsky.net

*************************************/
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <sys/time.h>
#include <unistd.h>
#include <stdio.h>

int main(void)
{
    int fd;
    struct input_event
    {
        struct timeval time;
        unsigned short type;
        unsigned short code;
        long value;
    } Point;
    fd = open("/dev/input/event1", 0, 0);                                   //Open button device
    if (fd < 0)
    {
        perror("open /dev/input/event1:");
        exit(1);
    }
    for (;;)
    {
        read(fd,&Point,sizeof Point);                                   //Capture button state information
        printf("Type: %d Code: %d \n", Point.type, Point.code);        //print out button information
    }
    return 0;
}
```

Cross-compile the test program by executing "arm-linux-gcc -o buttons buttons.c" or "make", and download the executable program "buttons" to platform to execute it.



7. 2 Example of driver development in Linux

Caution: Make sure the cross-compiler of version 3.4.1 has been installed.

Execute the command "# arm-linux-gcc -v". If the information above the red underline in the upper program, it indicates that the cross-compiler of version 3.4.1 has been installed; If there is no such information, please consult "2.5.1"

The location of the source code:

−The source code of experiment 1: /opt/EmbedSky/kernel-2.6.13/drivers/char/TQ2440_hello.c

−The source code of experiment 2: /opt/EmbedSky/kernel-2.6.13/ drivers/char/TQ2440_leds.c

−The source code of experiment 3: /opt/EmbedSky/kernel-2.6.13/ drivers/input/keyboard/TQ2440_buttons.c

7. 2. 1 Hello EmbedSky experiment

We provide an example in the following contents to illustrate the general steps of Linux driver development.

It is suggested to use the macro module_init() and module_exit() to record the initialization function and exit function.

There are 2 methods to load driver in Linux:

- In the process of system start-up, load driver module by code itself.
- After system start-up, load driver module by using the instructions "insmod" and so on.

The driver module is loaded or unloaded by system calling; Or the driver entry is put to a certain place when compiling, and a group of specific codes find the entry and load this driver after kernel start-up. The macro-definition "module_init()" and "module_exit()" are in the kernel source code file "include/linux/init.h".

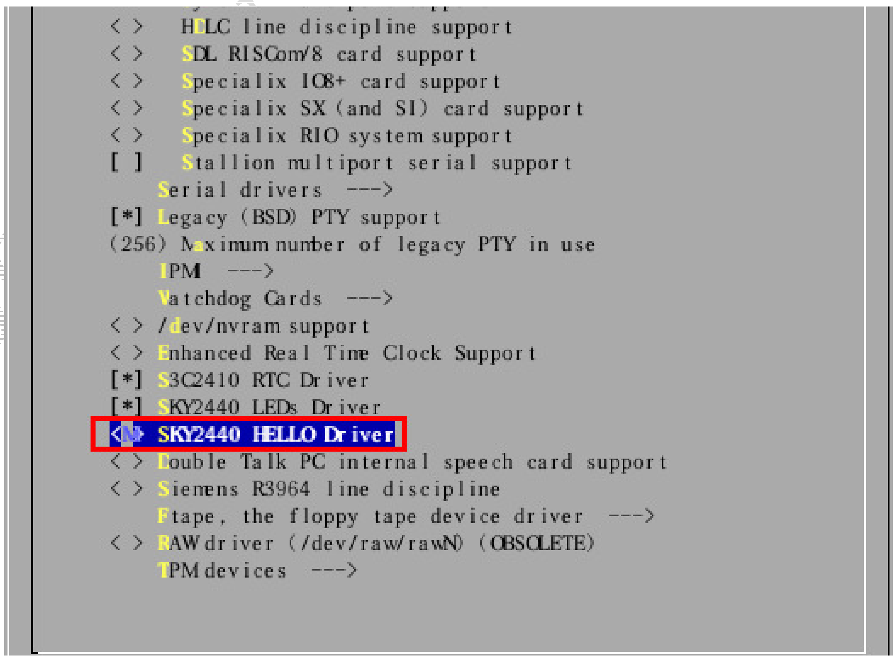
The following example program is used for calling print command to load and unload driver.

The source code:

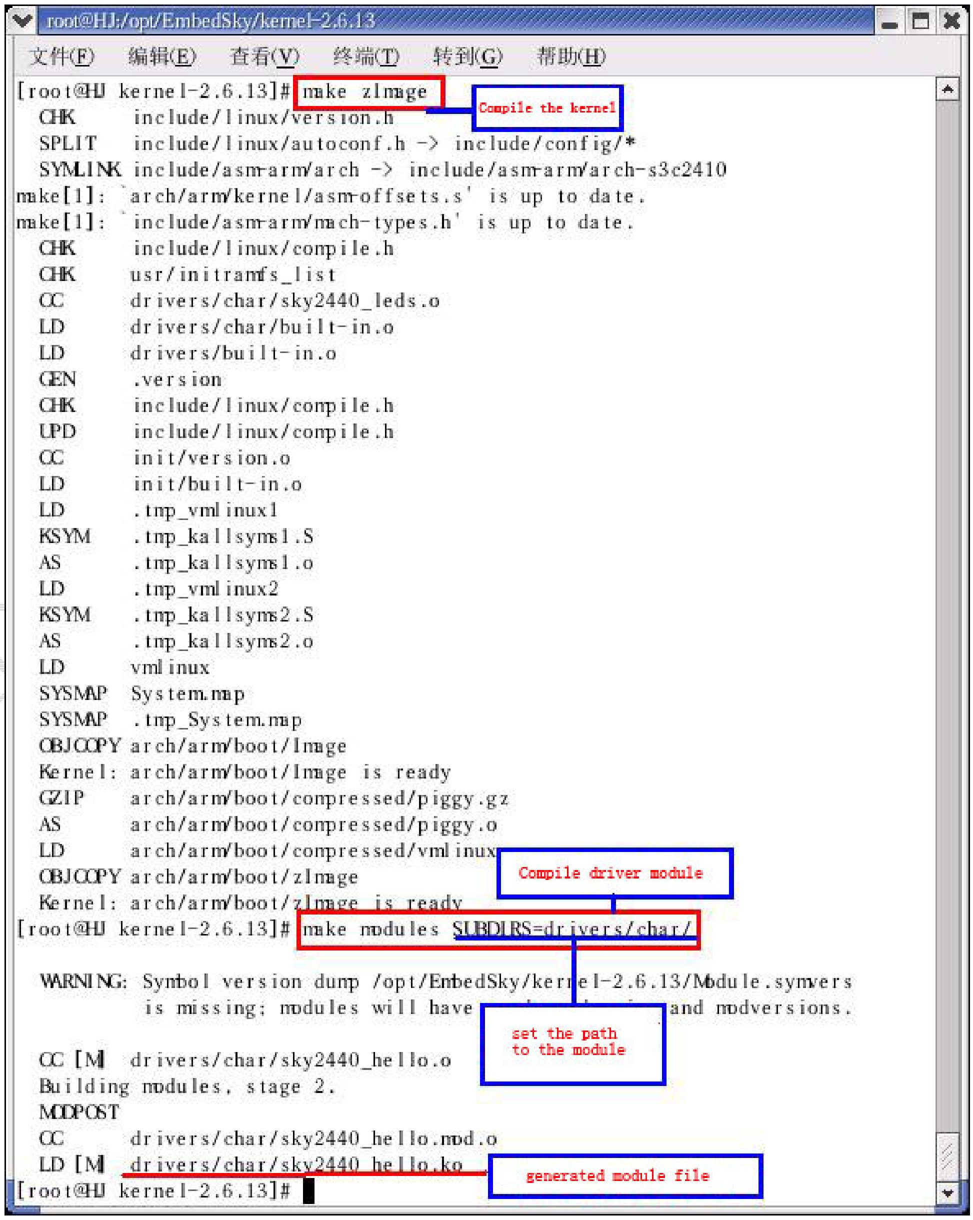
```
/*************************************

NAME:TQ2440_hello.c
COPYRIGHT:www.embedsky.net

*************************************/
#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/devfs_fs_kernel.h>
#include <linux/miscdevice.h>
#include <linux/delay.h>
#include <asm/irq.h>
#include <asm/arch/regs-gpio.h>
#include <asm/hardware.h>

MODULE_LICENSE("Dual BSD/GPL");

static int __init TQ2440_hello_init(void)                    //Driver initialization function
{
    printk("<1>\n       Hello,EmbedSky!\n");
    printk("<1>\nThis is first driver program.\n\n");
    return 0;
}

static void __exit TQ2440_hello_exit(void)                   //Driver exit function
{
```

```
    printk("<1>\n        Exit!\n");
    printk("<1>\nGoodbye EmbedSky!\n\n");
}

module_init(TQ2440_hello_init);                    //Driver module initialization macro
module_exit(TQ2440_hello_exit);                    //Driver module exit macro
```

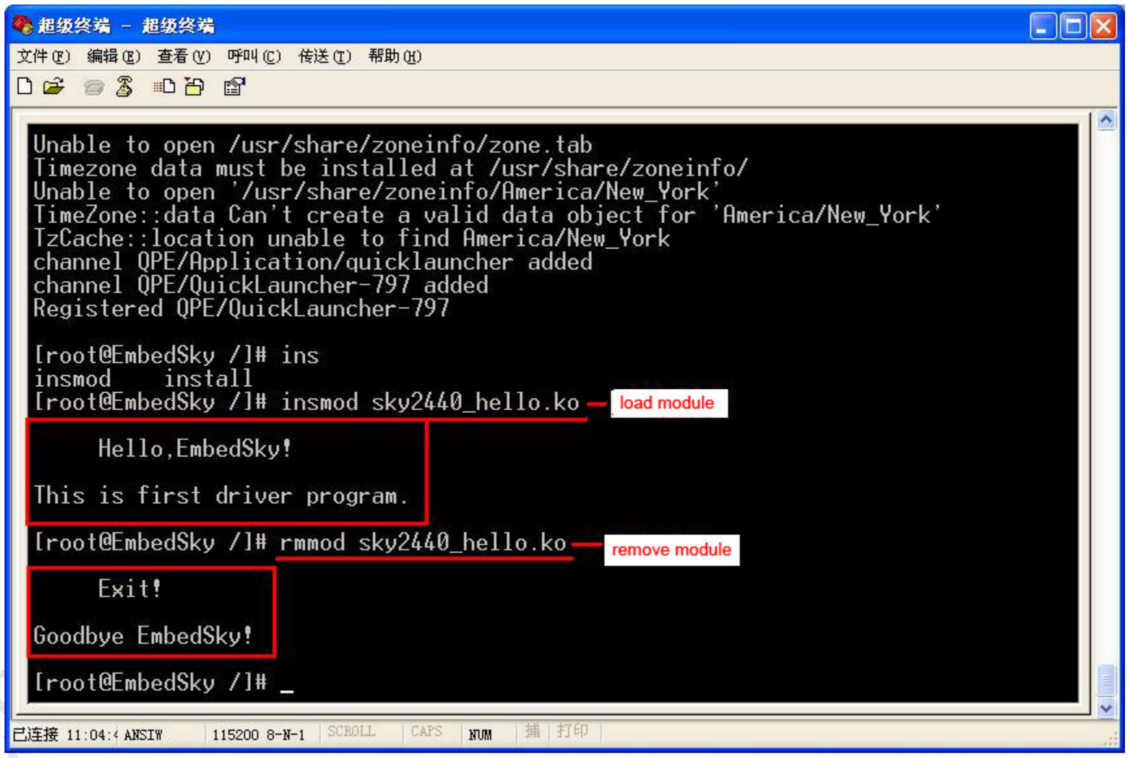
When configuring the kernel, select "M" for "TQ2440_HELLO" and set the path "Device Drivers->Character devices->TQ2440 HELLO Driver":



After configuration, compile the kernel first and then compile the module.



Download the compiled file “TQ2440_hello.ko” to platform:



7. 2. 2 Experiment of LED driver

The user LED driver controls the LED by controlling the 4 I/O ports of MCU. The following contents introduce the structure of character device driver program.

The source code of driver:

```
/*************************************

NAME:TQ2440_leds.c
COPYRIGHT:www.embedsky.net

*************************************/

#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/devfs_fs_kernel.h>
#include <linux/miscdevice.h>
#include <linux/delay.h>
#include <asm/irq.h>
#include <asm/arch/regs-gpio.h>
```

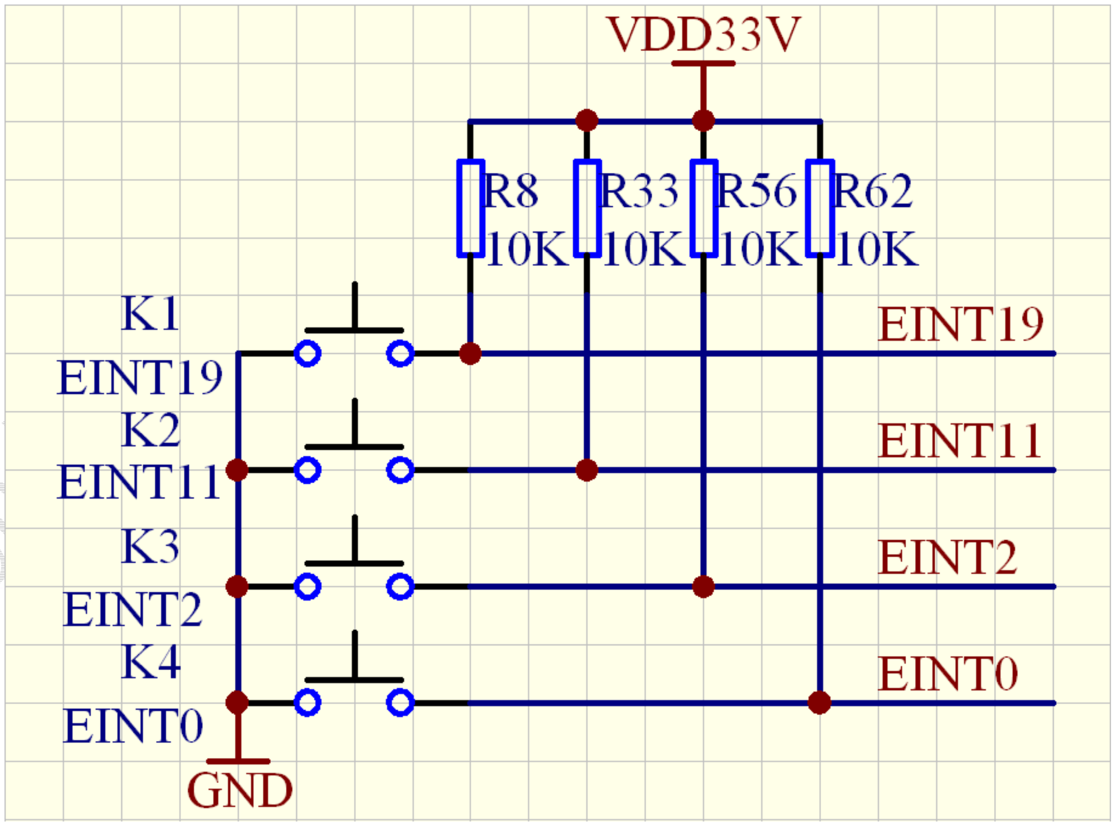
```
#include <asm/hardware.h>

#define DEVICE_NAME     "TQ2440_leds"                    //Define the device name
#define LED_MAJOR       231                              //Define the major device serial number

static unsigned long led_table [] =                      //The I/O mode corresponding to hardware resources
{
    S3C2410_GPB5,
    S3C2410_GPB6,
    S3C2410_GPB7,
    S3C2410_GPB8,
};

static unsigned int led_cfg_table [] =
{
    S3C2410_GPB5_OUTP,
    S3C2410_GPB6_OUTP,
    S3C2410_GPB7_OUTP,
    S3C2410_GPB8_OUTP,
};

static int TQ2440_leds_ioctl(struct inode *inode, struct file *file, unsigned int cmd, unsigned long arg)
                                                         //Use ioctl to control LED
{
    switch(cmd)
{
    case 0:
    case 1:
        if (arg > 4)
    {
            return -EINVAL;
        }
        s3c2410_gpio_setpin(led_table[arg], !cmd);
        return 0;
    default:
        return -EINVAL;
    }
}

static struct file_operations TQ2440_leds_fops =
{
    .owner  =   THIS_MODULE,
```

```
	.ioctl	=	TQ2440_leds_ioctl,
};

static int __init TQ2440_leds_init(void)
{
	int ret;
	int i;

	ret = register_chrdev(LED_MAJOR, DEVICE_NAME, &TQ2440_leds_fops);
							//Register the device in kernel
	if (ret < 0)
	{
		printk(DEVICE_NAME " can't register major number\n");
		return ret;
	}

	devfs_mk_cdev(MKDEV(LED_MAJOR, 0), S_IFCHR | S_IRUSR | S_IWUSR | S_IRGRP,
DEVICE_NAME);

	for (i = 0; i < 4; i++)
	{
		s3c2410_gpio_cfgpin(led_table[i], led_cfg_table[i]);
		s3c2410_gpio_setpin(led_table[i], 1);
	}

	printk(DEVICE_NAME " initialized\n");
	return 0;
}

static void __exit TQ2440_leds_exit(void)
{
	devfs_remove(DEVICE_NAME);
	unregister_chrdev(LED_MAJOR, DEVICE_NAME);
}

module_init(TQ2440_leds_init);
module_exit(TQ2440_leds_exit);
```

Select “Y” in the option “TQ2440_LED” and set the path “Device Drivers->Character devices->TQ2440 LED Driver”. The process of compiling is the same with compiling “TQ2440_HELLO” introduced previously. The LED driver has been included in kernel image in CD-ROM, and the user could use the LED controlling application compiled previously to operate the driver.

7. 2. 3 Experiment of user keyboard driver

The control mode of keyboard on platform is interrupt. The 4 user button are corresponding to the interrupt sources 0, 2, 11 and 19 (please consult the schematic diagram in CD-ROOM).



The steps of using the external interrupt:

- Register interrupt function, and call it when interrupt occurs.
- Capture the voltage level of pin when interrupt occurs.
- Clear the interrupt state.

The source code:

```
/*************************************

NAME:TQ2440_buttons.c
COPYRIGHT:www.embedsky.net

*************************************/

#include <linux/config.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/input.h>
```

```
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/interrupt.h>
#include <linux/device.h>
#include <asm/io.h>
#include <asm/irq.h>
#include <asm/arch/regs-gpio.h>

#define TQ2440_BUTVERSION  0x0001
#define DEVICE_NAME             "TQ2440-buttons"//Device name

#define REPEAT_DELAY            HZ/10

#ifdef DEBUG
#define dprintk(msg...) printk(KERN_DEBUG "TQ2440_buttons: " msg);
#else
#define dprintk(msg...)
#endif

MODULE_AUTHOR("Yellow <think3133@yahoo.com.cn>");
MODULE_DESCRIPTION("TQ2440 buttons Driver");
MODULE_LICENSE("GPL");

struct TQ2440_button {
	int irq;
	int pin;
	int pin_setting;
	int keycode;
	char *name;
	int last_state;
	struct timer_list timer;
};

static struct TQ2440_button TQ2440_buttons[] = //Device structure
{
	{ IRQ_EINT0,   S3C2410_GPF0,   S3C2410_GPF0_EINT0,    KEY_3,    "TQ2440_K1", 0 },
	{ IRQ_EINT2,   S3C2410_GPF2,   S3C2410_GPF2_EINT2,    KEY_2,    "TQ2440_K2", 0 },
	{ IRQ_EINT11, S3C2410_GPG3,   S3C2410_GPG3_EINT11,   KEY_1,    "TQ2440_K3", 0 },
	{ IRQ_EINT19, S3C2410_GPG11, S3C2410_GPG11_EINT19,  KEY_ESC,  "TQ2440_K4", 0 },
};

struct TQ2440_buttons_private           //Private device data structure
{
```

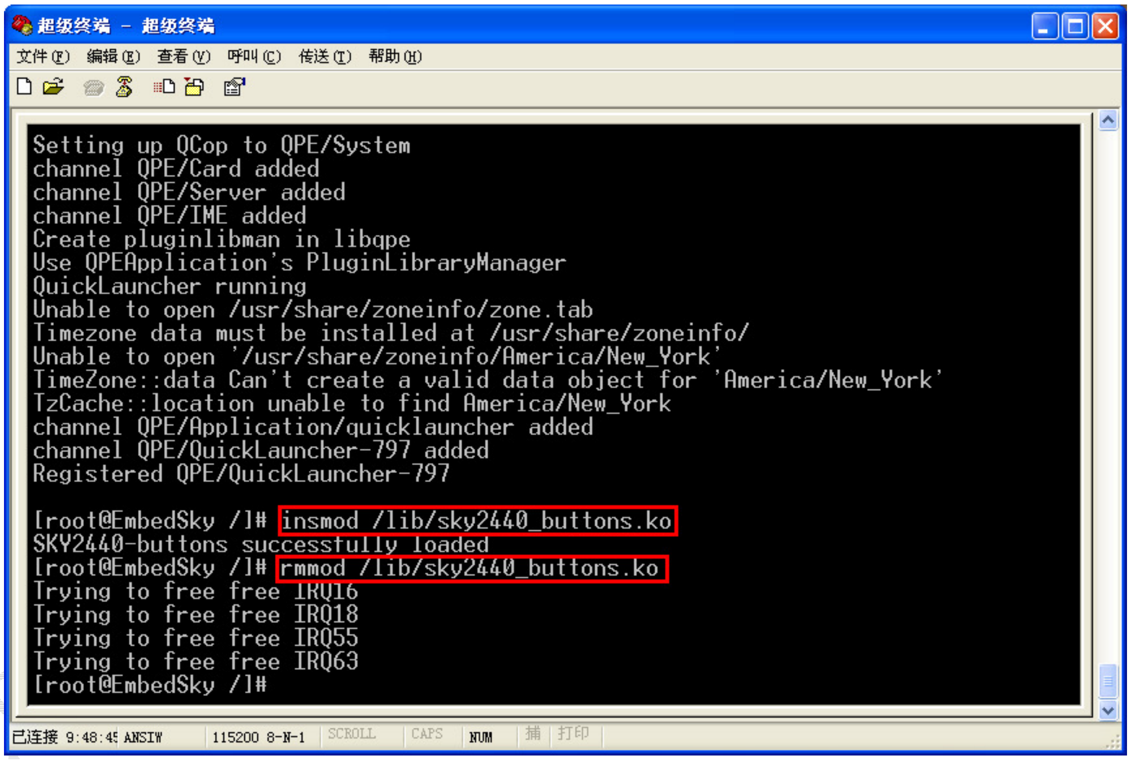
```
    struct input_dev      dev;
    spinlock_t            lock;
    int                   count;
    int                   shift;
    char                  phys[32];
};

static struct TQ2440_buttons_private priv;

static irqreturn_t TQ2440_buttons_keyevent(int irq, void *dev_id, struct pt_regs *regs)     //Keyboard    action
processing function
{
    struct TQ2440_button *button = (struct TQ2440_button *)dev_id;
    int down;

    if (!button)
        return IRQ_HANDLED;

    down = !(s3c2410_gpio_getpin(button->pin));

    if (button->last_state == down)
        return IRQ_HANDLED;

    button->last_state = down;

    dprintk("%s button %s\n",button->name, down ? "pressed" : "released");

    input_report_key(&priv.dev, button->keycode, down);
    input_sync(&priv.dev);

    if (down)
        mod_timer(&button->timer, jiffies + REPEAT_DELAY);

    return IRQ_HANDLED;
}

static void TQ2440_buttons_timer_callback(unsigned long data)
{
        struct TQ2440_button *button = (struct TQ2440_button *) data;
        int down;

        down = !(s3c2410_gpio_getpin(button->pin));
```

```
		if (down) {
			dprintk("Timer: %s button %s\n",button->name, down ? "pressed" : "released");
			input_report_key(&priv.dev, button->keycode, down);
			input_sync(&priv.dev);
			mod_timer(&button->timer, jiffies + REPEAT_DELAY);
		}
}

static int __init TQ2440_buttons_probe(struct device *dev)//Keyboard loading function
{
	int i;

	memset(&priv, 0, sizeof(struct TQ2440_buttons_private));
	init_input_dev(&priv.dev);
	priv.dev.evbit[0] = BIT(EV_KEY);
	sprintf(priv.phys, "input/TQ2440_buttons0");

	priv.dev.private = &priv;
	priv.dev.name = DEVICE_NAME;
	priv.dev.phys = priv.phys;
	priv.dev.id.bustype = BUS_HOST;
	priv.dev.id.vendor = 0xDEAD;
	priv.dev.id.product = 0xBEEF;
	priv.dev.id.version = TQ2440_BUTVERSION;

	for (i = 0; i < ARRAY_SIZE (TQ2440_buttons); i++) {
		set_bit(TQ2440_buttons[i].keycode, priv.dev.keybit);
		s3c2410_gpio_cfgpin(TQ2440_buttons[i].pin,TQ2440_buttons[i].pin_setting);
		request_irq (TQ2440_buttons[i].irq, TQ2440_buttons_keyevent,\
				SA_SAMPLE_RANDOM, TQ2440_buttons[i].name, &TQ2440_buttons[i]);
		set_irq_type(TQ2440_buttons[i].irq, IRQT_BOTHEDGE);

		init_timer(&TQ2440_buttons[i].timer);
		TQ2440_buttons[i].timer.function = TQ2440_buttons_timer_callback;
		TQ2440_buttons[i].timer.data	= (unsigned long)&TQ2440_buttons[i];
	}

	printk(KERN_INFO "%s successfully loaded\n", DEVICE_NAME);

	input_register_device(&priv.dev);

	return 0;
}
```

```
static int TQ2440_buttons_remove(struct device *dev) //Keyboard removing function
{
	int i;

	for (i = 0; i < ARRAY_SIZE (TQ2440_buttons); i++) {
		disable_irq(TQ2440_buttons[i].irq);
		free_irq(TQ2440_buttons[i].irq,&priv.dev);
	}

	input_unregister_device(&priv.dev);

	return 0;
}

static struct device_driver TQ2440_buttons_driver =	//Driver format structure
{
		.name			= DEVICE_NAME,
		.bus			= &platform_bus_type,
		.probe			= TQ2440_buttons_probe,
		.remove			= TQ2440_buttons_remove,
};

int __init TQ2440_buttons_init(void)
{
	return driver_register(&TQ2440_buttons_driver);/Load the driver
}

void __exit TQ2440_buttons_exit(void)
{
	driver_unregister(&TQ2440_buttons_driver);	//Unload the driver
}

module_init(TQ2440_buttons_init);
module_exit(TQ2440_buttons_exit);
```

The kernel configuration list and compiled kernel in CD-ROM do not contain "TQ2440_buttons" driver, however, it could be modified. The file system "root_nfs" has been added with compiled keyboard module. And the path is "/opt/EmbedSky/root_nfs/lib/sky_buttons.ko".

Compiling method: Execute the command "make modules SUBDIRS=drivers/input/keyboard" after kernel compiling, and the module "sky_buttons.ko" is created under the directory "drivers/input/keyboard". And then download this module to file system to support mounting and unmounting:



The processes of loading and unloading are shown in the upper diagram.

3 methods of driver editing are introduced in the upper contents. Please pay more attention to the functions registering driver and uninstalling driver.

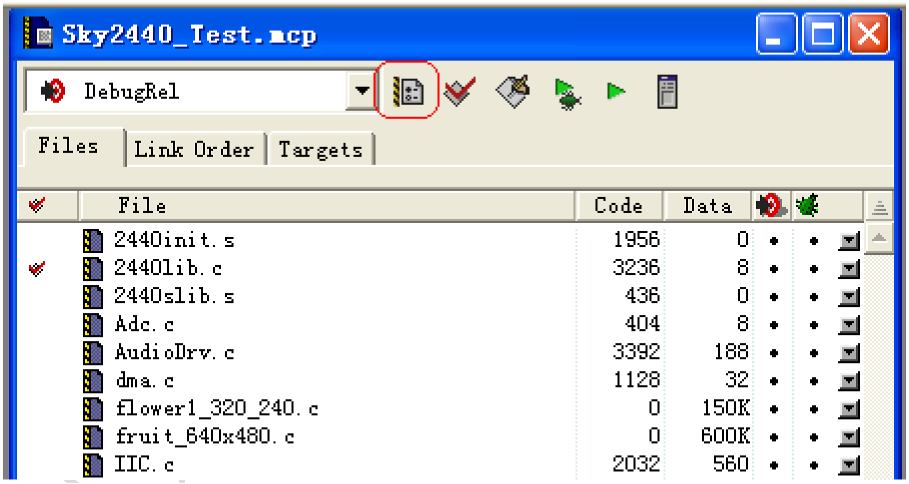
7. 3 Experiment of Non-OS application

This experiment is based on the integrated development environment ADS 1.2.

7.3.1 Configuring the experiment environment

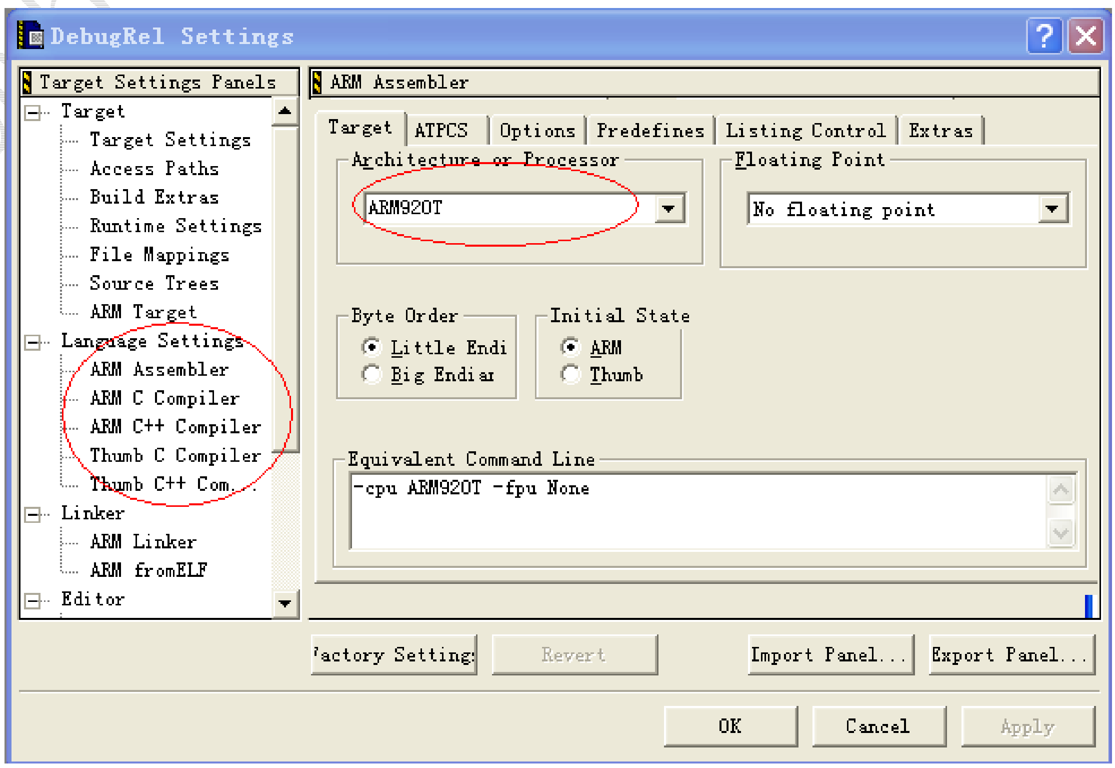
The configuration steps:

Step1, decompress the test application: enter into the directory and open the project "TQ2440_test.mcp" by using ADS:

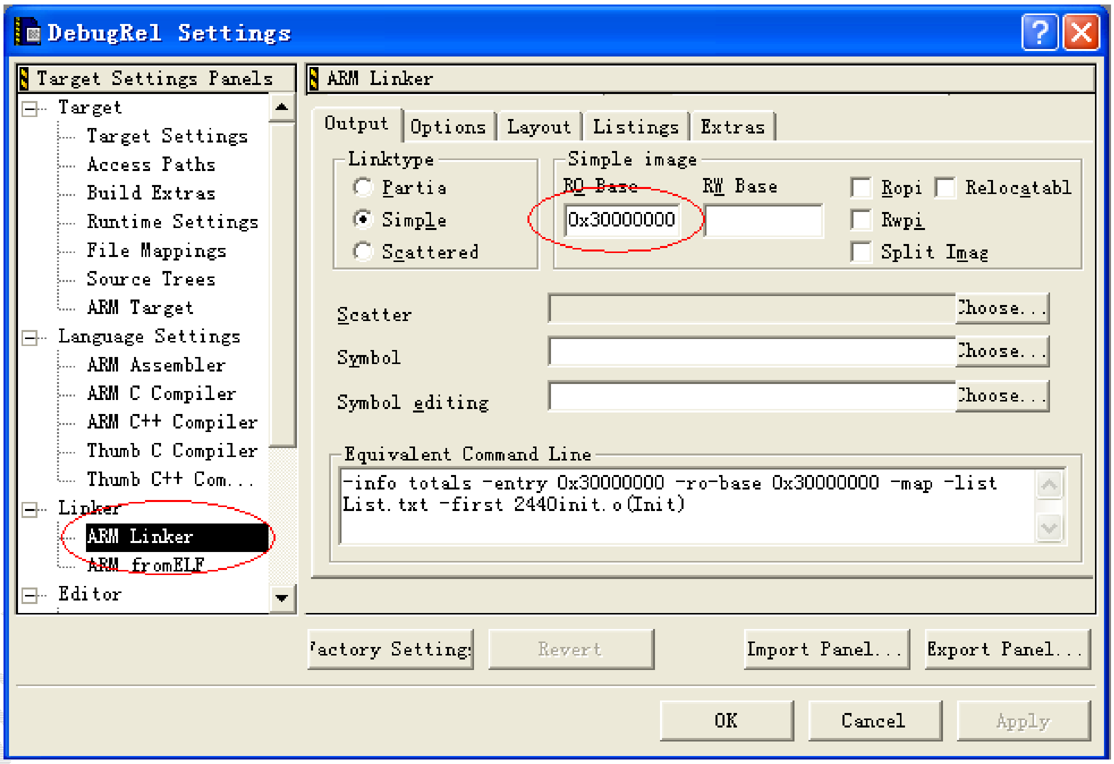


Step2, click the icon in red circle in the upper diagram or press “Alt+F7” to enter into the interface “Release settings” (or select “DebugRel” as target to enter into “DebugRes Settings”), or enter in from “Edit”:

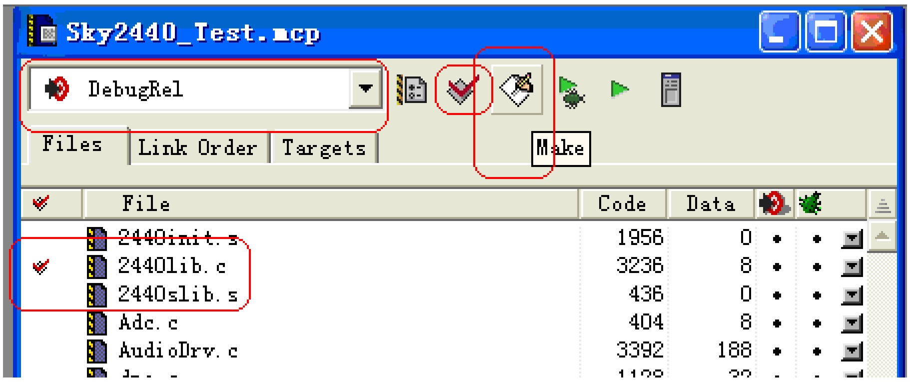
Check the settings in “Language Settings”: “Architecture or Processor” is “ARM920T” or not:



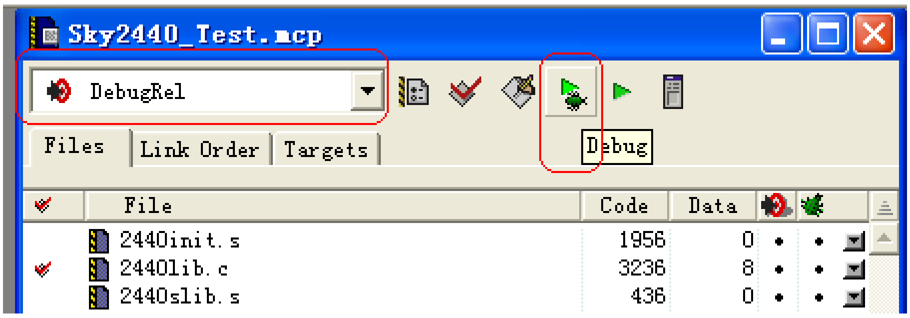
Check Linker: “ARM Linker”---“R0 base” is 0x30000000 or not:



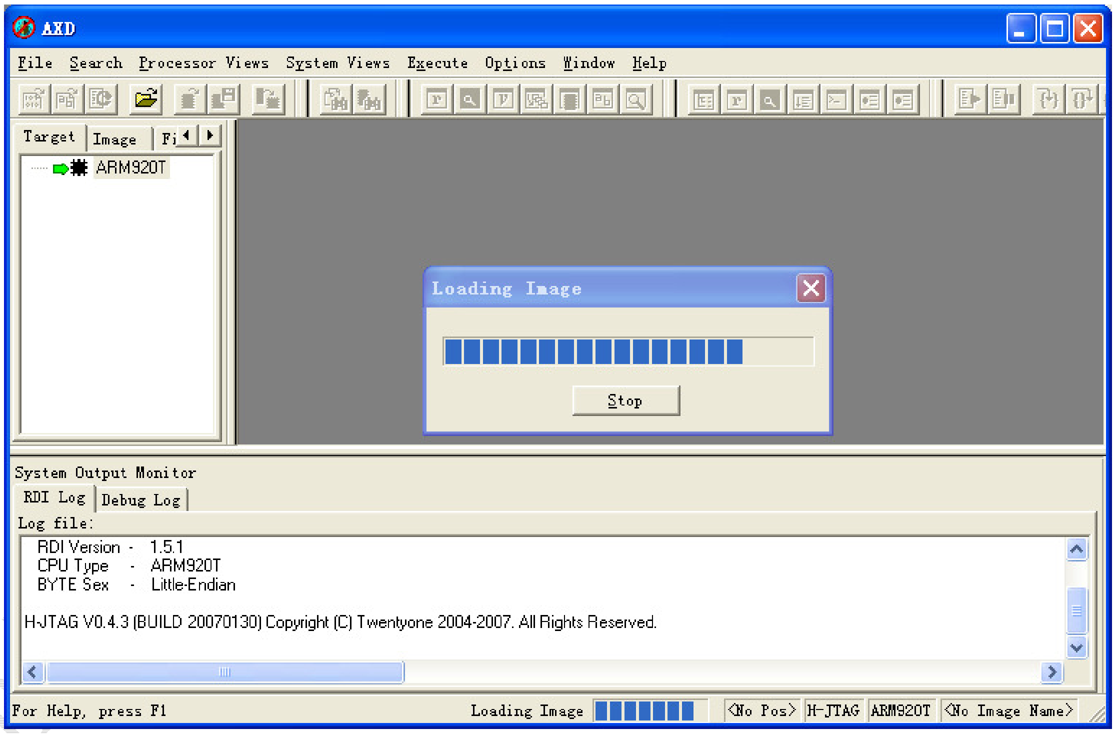
Step3, after the check, click the file with a prefixed mark (this mark is auto-added), and click “make” to start compiling. If there is no modification, skip this step. Caution: select “Target” mode “DebugRel”:



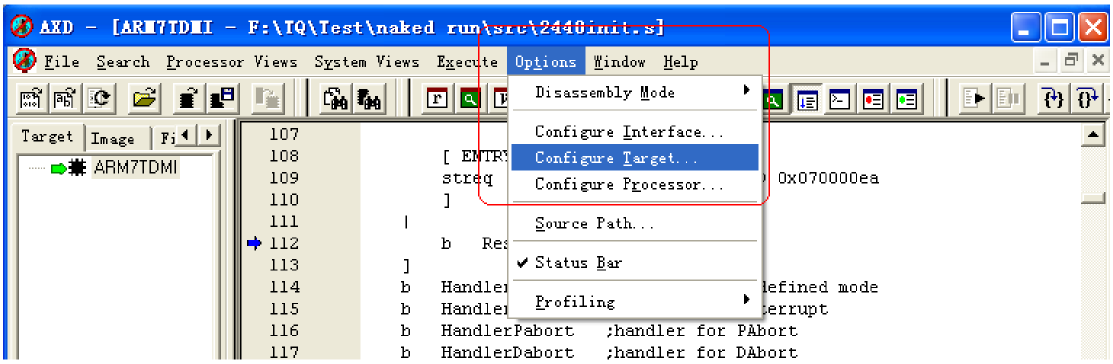
Step4, after compiling, finish the connection of the target board (Jtag, power and serial port). Start up the board and then enter into bootloader download mode; Start up H-JTAG and H-JTAG and auto-detect CPU (if no CPU is detected, please check the configuration of H-JTAG); Click “Debug” after detecting CPU:



The process of loading image:

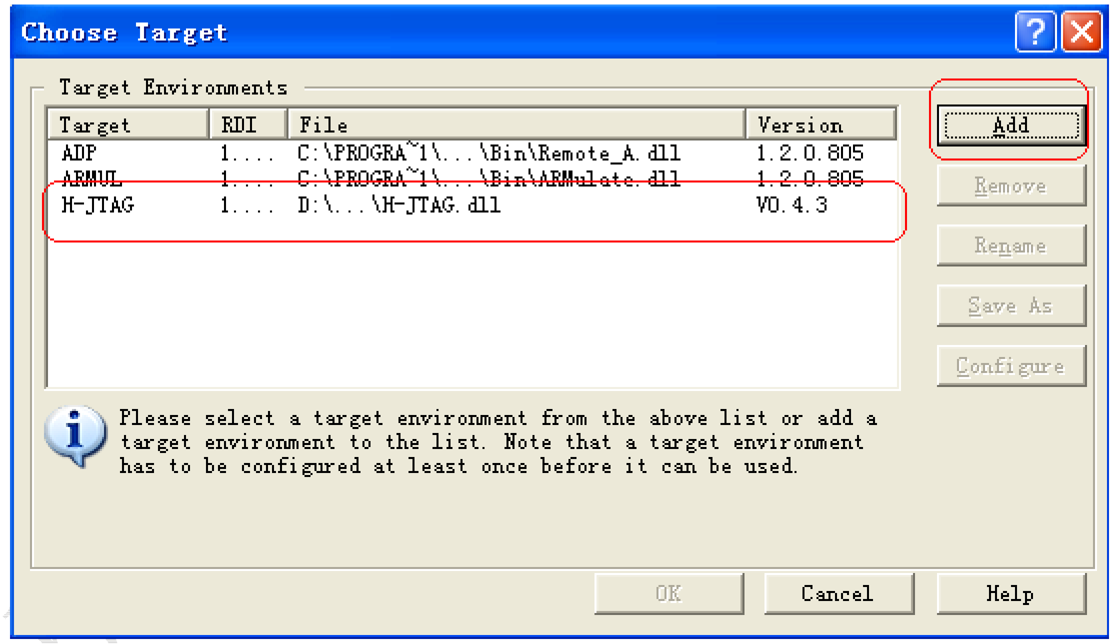


Step5, after loading image is finished, enter into the following AXD interface. Select “Options” --- “Configure target” to start configuring:



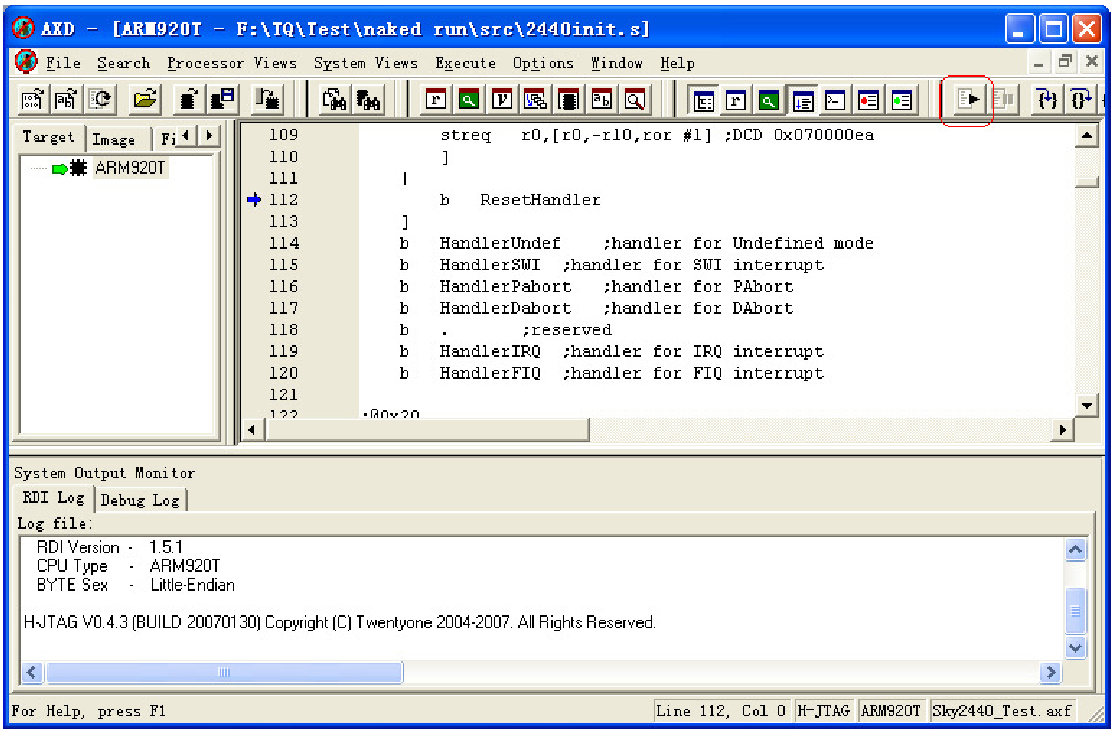
Step6, click “Add” to add “H-JTAG.DLL” (“H-JTAG.DLL” is under the folder “H-JTAG”). Select “ADP, ARMUL” and click “Remove” to remove it. And click “OK” to finish the configuration.

Caution: The configuration is needed only in the first use.

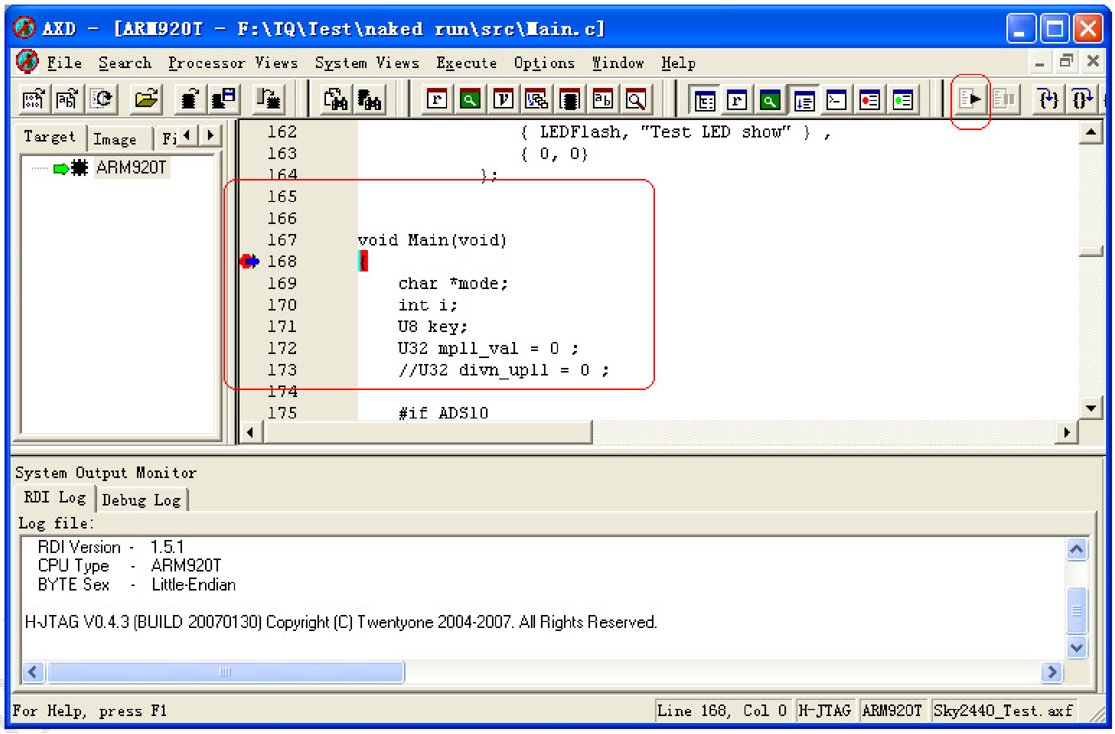


7.3.2 Experiment of test

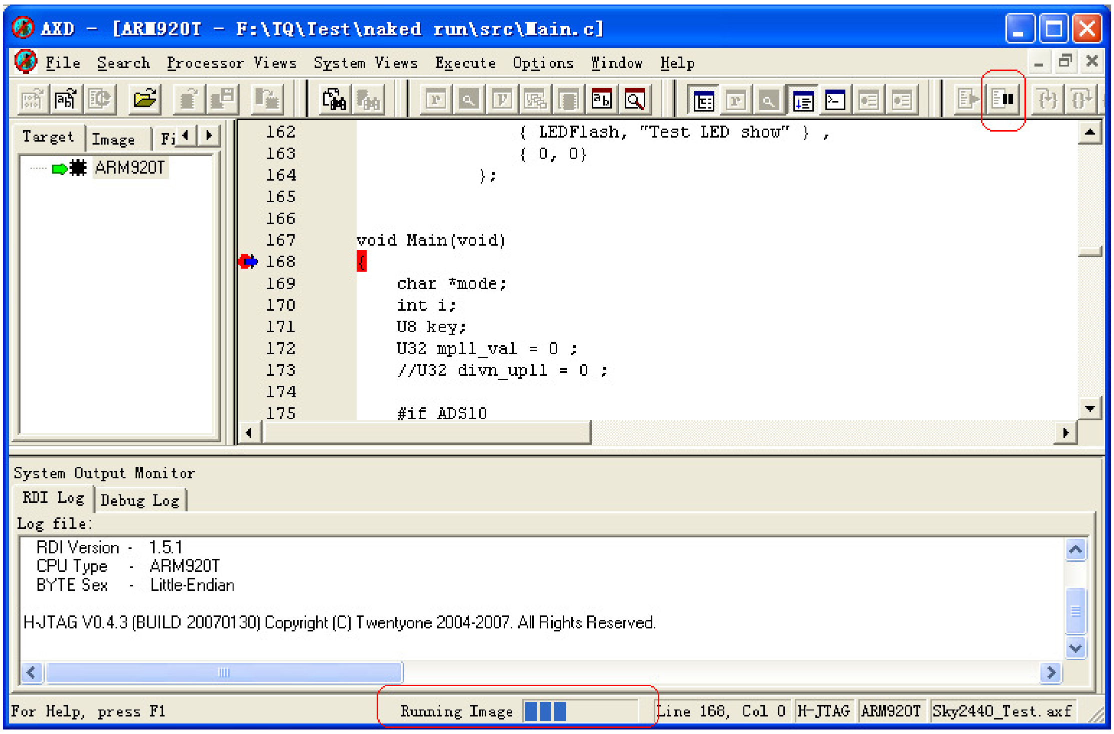
Step1, click the icon "go" in the following red circle or click "go" twice in the menu "Execute". Run the assembler:



Click "go" to run the main program:



Step2, the test program starts to run:



Step3, switch to hyper terminal, the main menu appears, as shown in the following diagram.

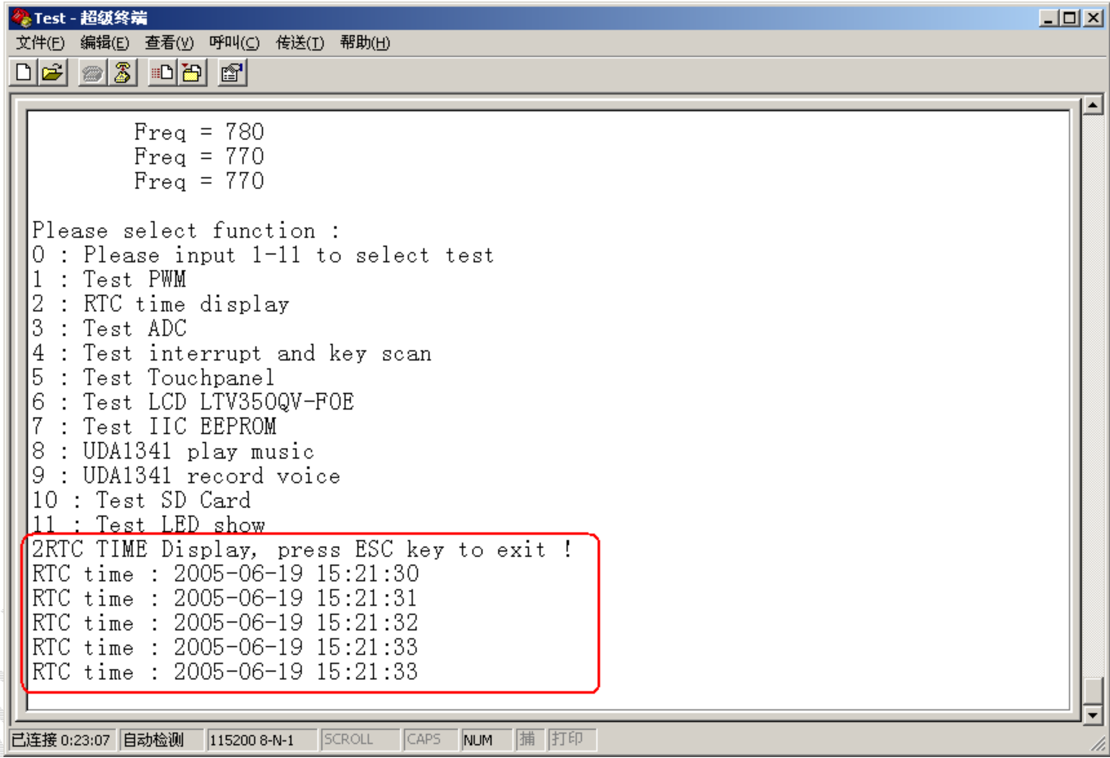
Select “0”, the main menu appears again; If the LCD has already been connected, the six color stripes “blue, green, red, yellow, pink and viridity” and “米” shape black line appear on LCD.

```
[7] Download to SDRAM & Run
[8] Boot the system
[9] Format the Nand Flash
Enter your selection:
<**************************************************>
           SKY2440 Test Program VER1.0
                www.embedsky.net
      Build time is: Nov 13 2007  23:38:07
<**************************************************>

Please select function :
0 : Please input 1-11 to select test
1 : Test PWM
2 : RTC time display
3 : Test ADC
4 : Test interrupt and key scan
5 : Test Touchpanel
6 : Test LCD LTV350QV-F0E
7 : Test IIC EEPROM
8 : UDA1341 play music
9 : UDA1341 record voice
10 : Test SD Card
11 : Test LED show
```

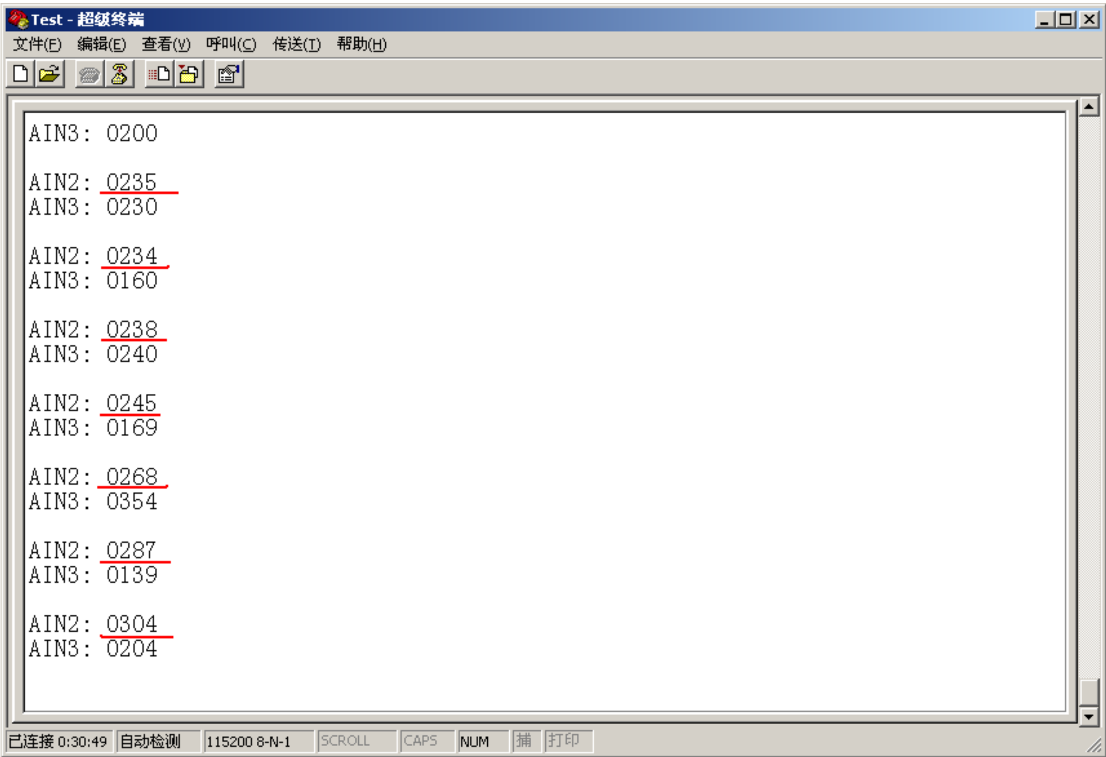
Step4, select “1”, the PWM display starts to run, press “+/-” (caution: Use the keypad, because the key “shift + ” is not supported.) to increase and decrease the frequency; Press “ESC” to exit and go back to the main menu:

```
3 : Test ADC
4 : Test interrupt and key scan
5 : Test Touchpanel
6 : Test LCD LTV350QV-F0E
7 : Test IIC EEPROM
8 : UDA1341 play music
9 : UDA1341 record voice
10 : Test SD Card
11 : Test LED show
1
BUZZER TEST ( PWM Control )
Press +/- to increase/reduce the frequency of BUZZER !
Press 'ESC' key to Exit this program !

        Freq = 810
        Freq = 820
        Freq = 830
        Freq = 820
        Freq = 810
        Freq = 800
        Freq = 790
        Freq = 780
        Freq = 770
```

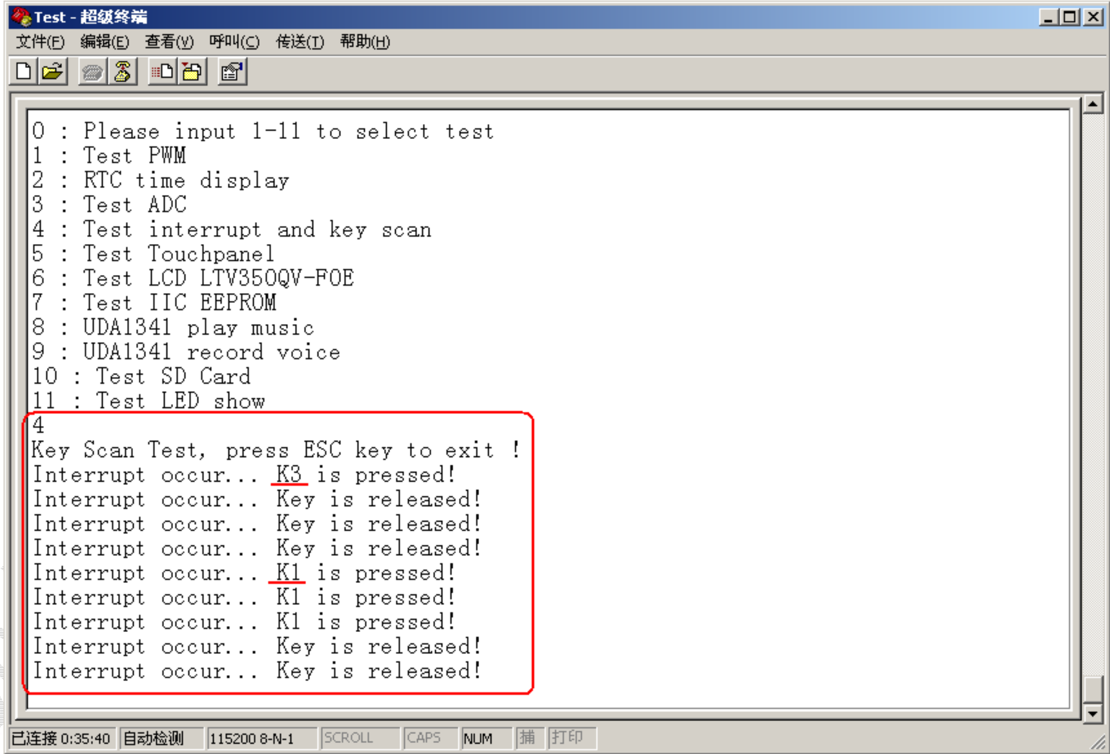
Step5, select “2” to run RTC test. It appears that the timer keeps increasing; Press “ESC” to exit and go back to the main menu:



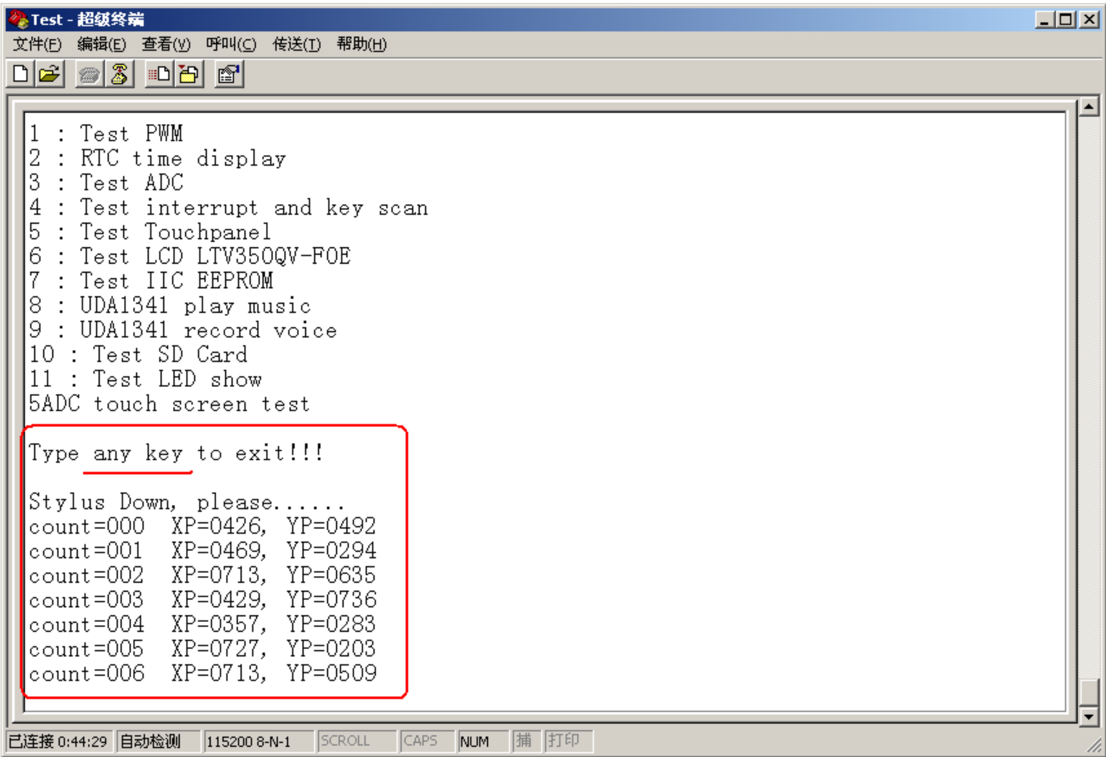
Step6, select “3” to start ADC button test. Adjust the resistance and the displaying data changes correspondingly. Press “ESC” to exit and go back to the main menu: (the following diagram is oriented to adjusting the resistance “ain2”)



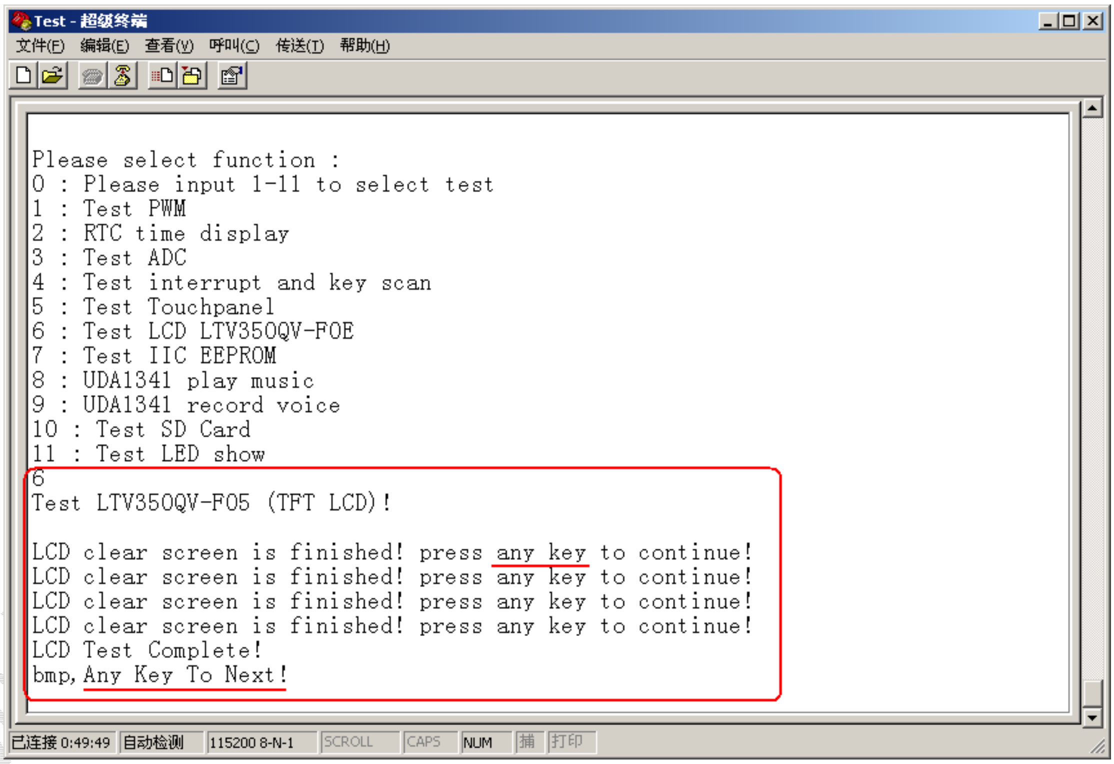
Step7, select “4” to start button test. The relevant testing information would appear. Press “ESC” to exit and go back to the main menu:



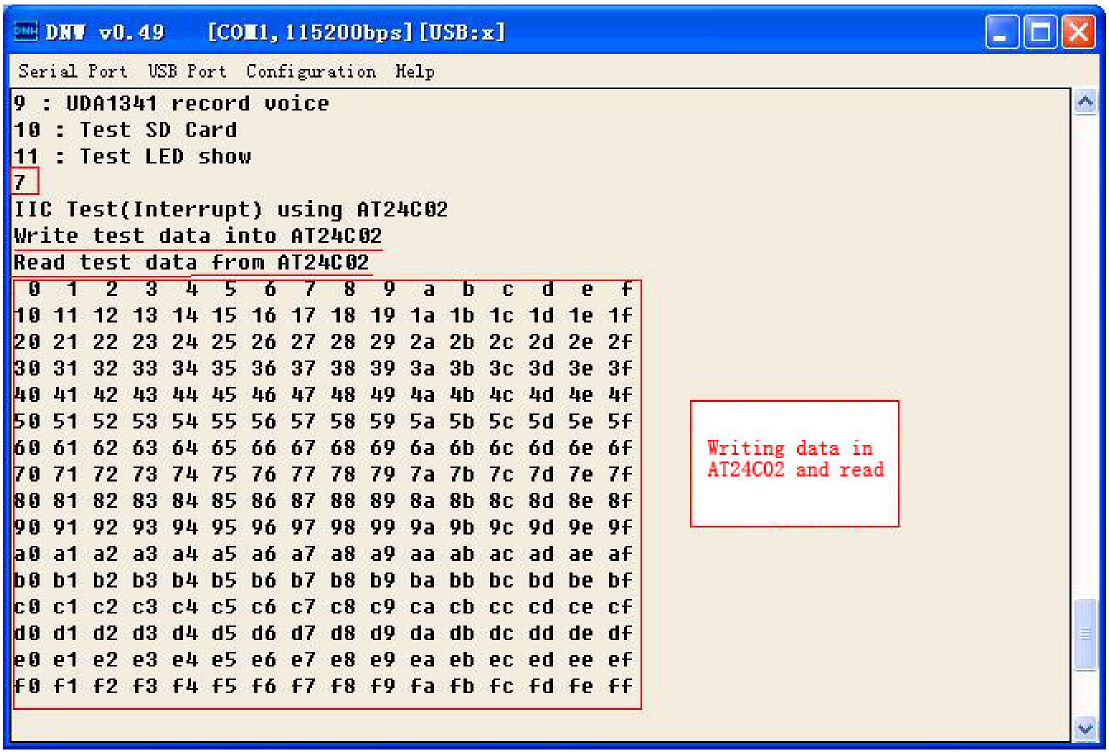
Step8, select “5” to start the touch screen test (connect the touch screen before the test). Touch the screen with a pen and the following information appears. Press any key to go back to the main menu:



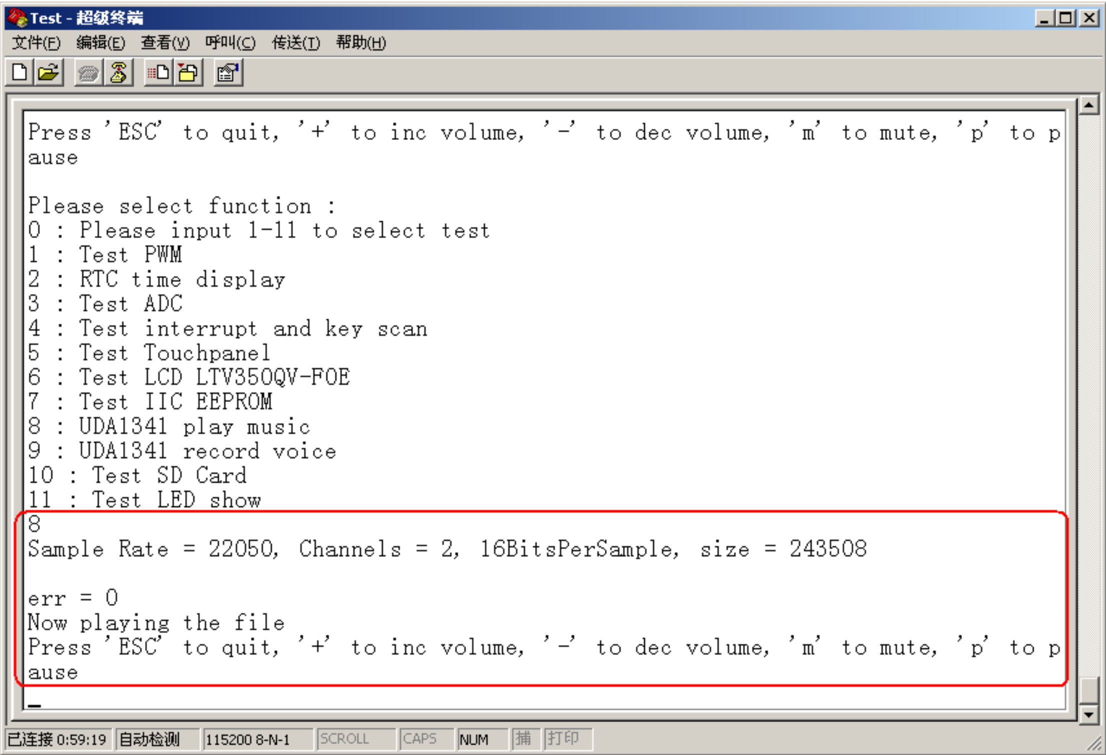
Step9, select “6” to start LCD test. The TFT screen would display the pictures “black ➔ white ➔ blue ➔ green ➔ 6 color stripes ➔ flower ➔ TQ logo”. Press any key to go back to the main menu.



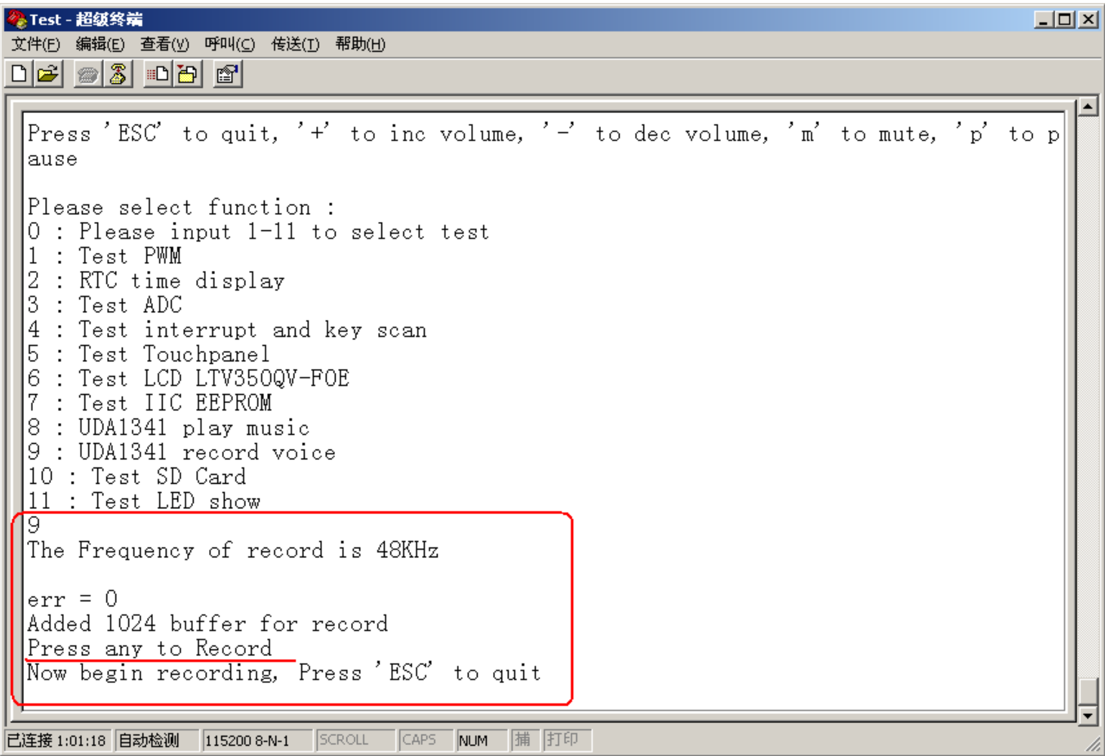
Step10, select “7” to start IIC test. CPU writes some data to IIC and then reads them. LCD has been cleared in selection “6”, and the 6 stripes would no longer appear:



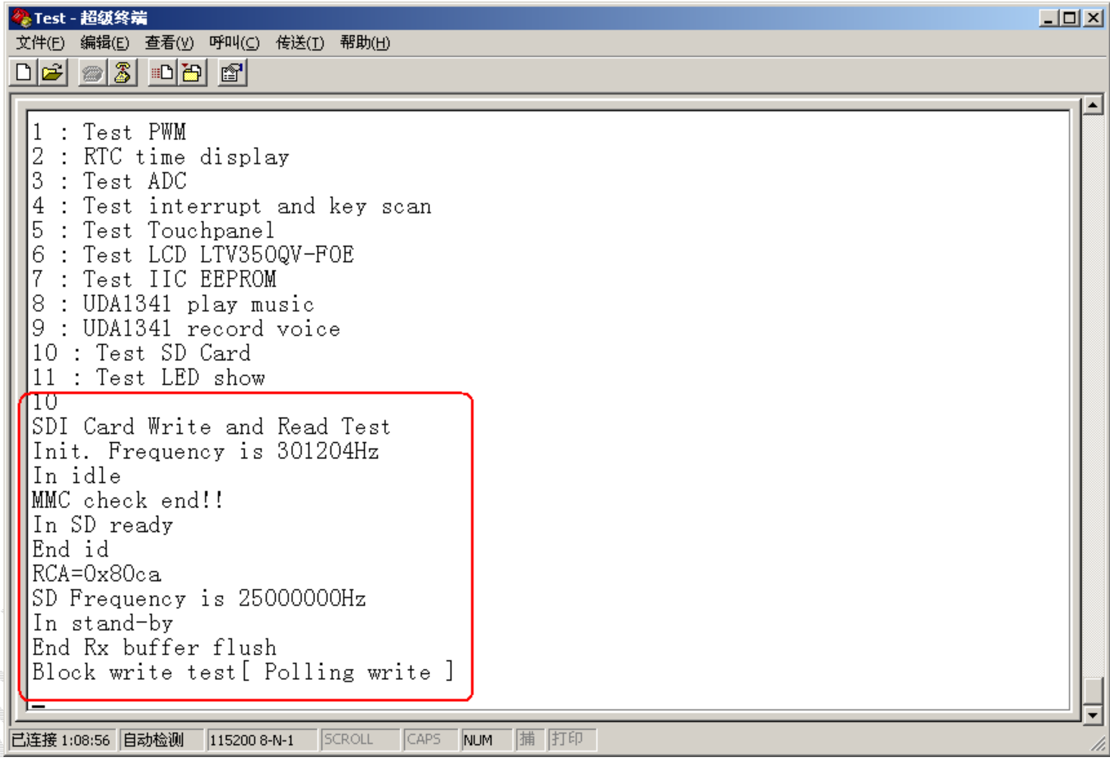
Step11, select “8” to start sound card test. Press “+” to increase the volume; “-” to decrease the volume; “m” to mute; “p” to pause. Press “ESC” to go back to the main menu:

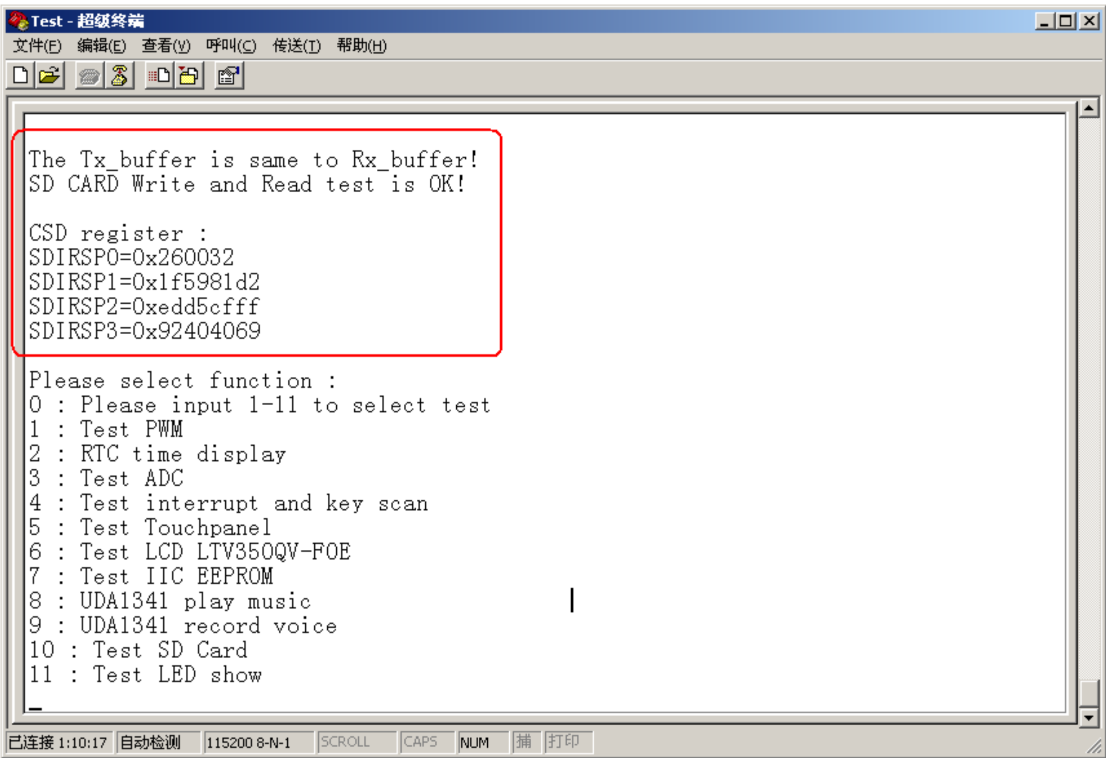


Step12, select “9” to start record test. Press any key to start. Press “ESC” to go back to the main menu:



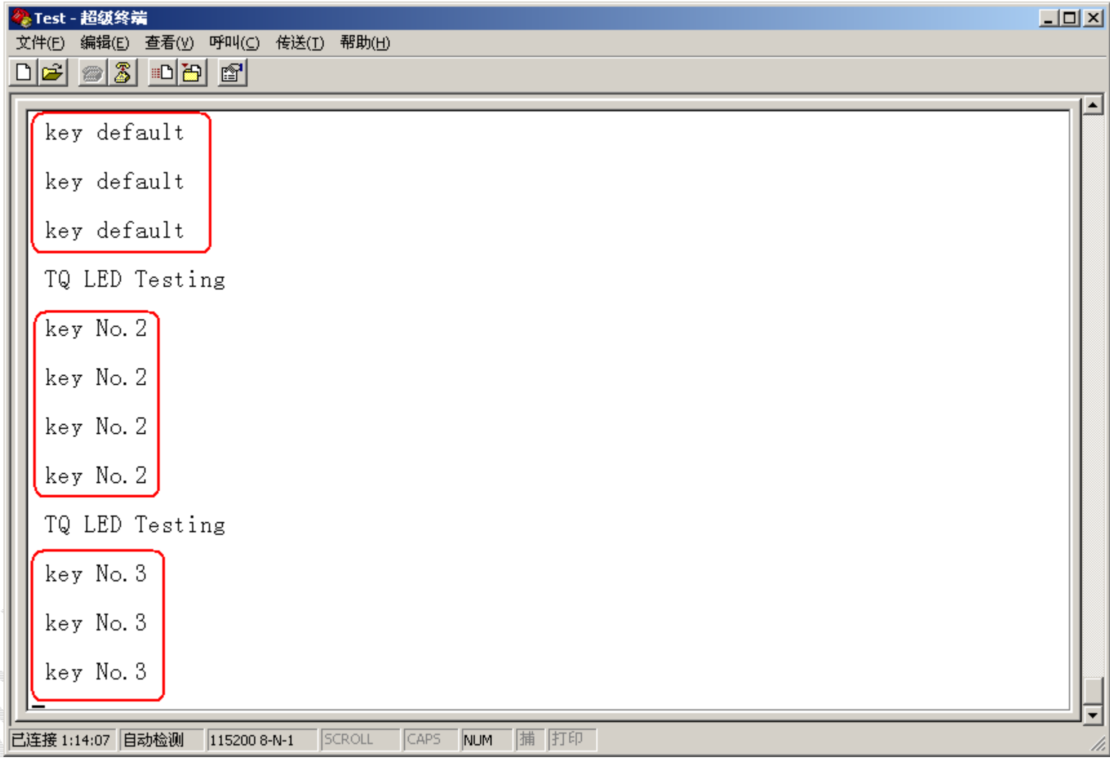
Step13, select “10” to start SD card test. Insert the SD card before the test begins. After the SD card information has been checked. It auto-goes back to the main menu:





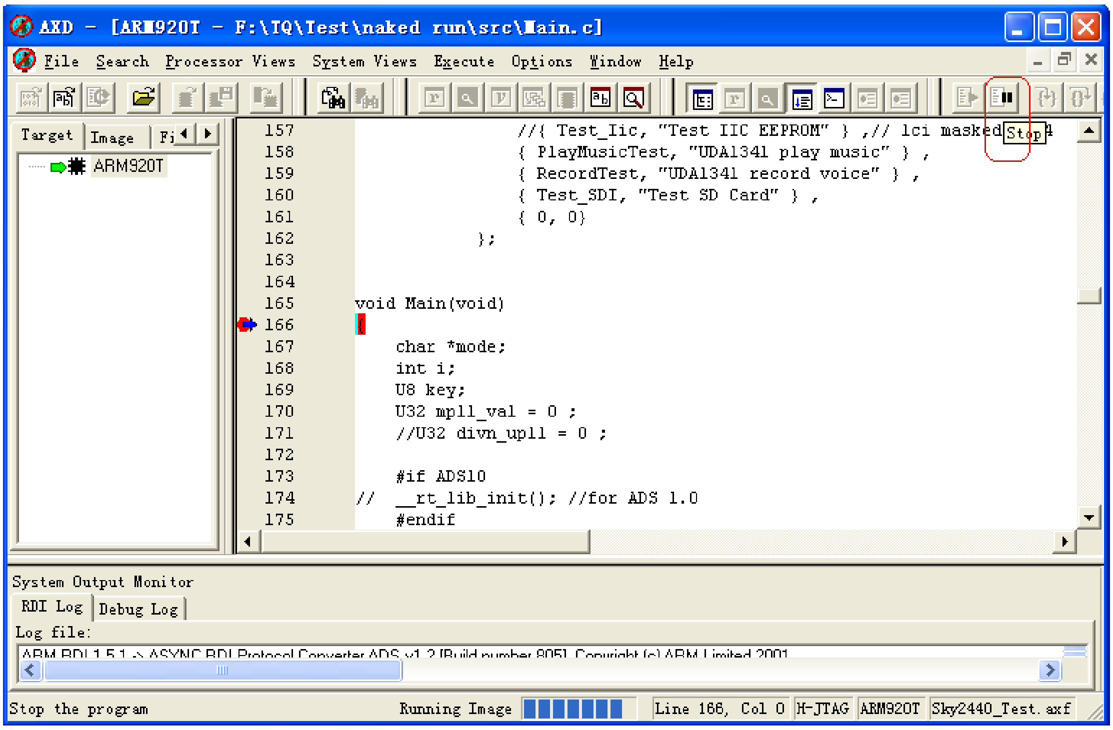
Step14, select “11” to start LED button controlling test. Different buttons are corresponding to different LED display mode. Press “ESC” to go back to the main menu.

Caution: The response of the interrupt is a bit slow, please press button and hold for a while during the test.



Step12, before stop the test, AXD must be stopped first. Click the icon “Stop” or select “Stop” in the menu “Execute”:

After the upper operation can user stop the AXD, terminal and the target board:



7. 3. 3 Burning TQ2440-Test.bin file

Please consult “2.4 节” to burn the program into Nor Flash, or consult the “Step7” in “2.6.3 节” to burn the program into SDRAM, and then jump to the relevant address to begin to run.

Chapter 8 Transplantation of Web server

The Web server used in SKY2410 is made up of "boa" and "cgic".

The building steps:

8. 1 Transplanting boa software

8. 1. 1 Building compiling environment

The official web site of "boa" is www.boa.org. The download address is: https://sourceforge.net/project/showfiles.php?group_id=78. The latest version is: boa-0.94.13

After download, decompress the file to the directory "/opt/EmbedSky/", and the directory "boa-0.94.13" is auto-created:

#tar xvfz boa-0.94.13.tar.gz -C /opt/EmbedSky/

8. 1. 2 Configuring the compiling condition

Configure "boa"：

#cd /opt/EmbedSky/boa-0.94.13/src

#./configure

Instruction: The prompt of configuration script about how to configure is not correct. So we use this method.

The Makefile is auto-created under "boa-0.94.13/src". Modify the Makefile:

#vi Makefile

In line 31 and line 32, find "CC = gcc 和 CPP = gcc -E". Modify this sentence into "CC = arm-linux-gcc 和 CPP =arm-linux-g++ -E". Save the modification and exit.

Modify the file "boa.c":

#vi boa.c

Find the following contents between line 225 and line 227, and annotate it:

```
if (setuid(0) != -1) {
        DIE("icky Linux kernel bug!");
    }
```

Save and exit.

8. 1. 3 Compiling and optimizing

After compiling , a executable file "boa" (about 232K) is auto-created under the directory "boa-0.94.13":

#make

Optimize the file:

#arm-linux-strip boa

The process of optimizing removes the debug information from boa, and the size changes from 232K to 62K.

The transplantation of is now complete.

8. 2 Transplanting cgic library

8. 2. 1 Building compile environment

The download web site of cgic library is: http://www.boutell.com/cgic/cgic205.tar.gz, and the latest version is cgic205 version

Download the file and decompress it to the directory "/opt/EmbedSky/", and the directory "cgic205" is auto-created:

#tar xvfz cgic205.tar.gz -C /opt/EmbedSky/

8. 2. 2 Configuring compile condition

Modify the file "Makefile" in the directory "cgic205":

#cd /opt/EmbedSky/cgic205

#vi Makefile

The contents after modification:

```
CFLAGS=-g -Wall
CC=arm-linux-gcc                    //Used to be CC = gcc
AR=arm-linux-ar                     //Used to be AR = ar
RANLIB=arm-linux-ranlib             //Used to be RANLIB = ranlib
LIBS=-L./ -lcgic

all: libcgic.a cgictest.cgi capture

install: libcgic.a
cp libcgic.a /usr/local/lib
cp cgic.h /usr/local/include
@echo libcgic.a is in /usr/local/lib. cgic.h is in /usr/local/include.

libcgic.a: cgic.o cgic.h
rm -f libcgic.a
$(AR) rc libcgic.a cgic.o
$(RANLIB) libcgic.a

#mingw32 and cygwin users: replace .cgi with .exe

cgictest.cgi: cgictest.o libcgic.a
```

$(CC) $(CFLAGS) cgictest.o -o cgictest.cgi ${LIBS}   //Change "gcc" into: $(CC) $(CFLAGS)

capture: capture.o libcgic.a
$(CC) $(CFLAGS) capture.o -o capture ${LIBS}   //Change "gcc" into: $(CC) $(CFLAGS)

clean:
rm -f *.o *.a cgictest.cgi capture
修改后保存退出。

8. 2. 3 Compiling and optimizing

After compiling , the executable file "capture" and test file "cgictest.cgi" is auto-created:
#make
Optimizing:
#arm-linux-strip capture
Decrease the size of "capture" from 100K to 29K.

8. 3 Configuring Web server

After the former transplantation works, now begin to configure Web server. (take NFS for example here)

8. 3. 1 Configuring boa

Create a directory named "web/", and create a directory named "boa/" under "etc" in file system:
#cd /opt/EmbedSky/root_nfs
#mkdir web etc/boa
Copy the transplanted file "boa" to the directory "sbin/" in file system:
#cp /opt/TQ/boa-0.94.13/src/boa /opt/TQ/root_nfs/sbin
Copy the boa configuration file "boa.conf" from the directory "boa-0.94.13" to "etc/boa/" in file system:
#cp /opt/EmbedSky/boa-0.94.13/boa.conf /opt/EmbedSky/root_nfs/etc/boa
Modify the file "boa.conf". (we give the contents modified and the approximate position here)
#cd /opt/EmbedSky/root_nfs/etc/boa
#vi boa.conf
The modified contents:
Port 80   //Line 25
//Listening port number, 80 is the default value.

#Listen 192.168.1.6   //Line 43
//The IP address used for binding, always be annotated. Means it is bound to INADDR_ANY, and it suit to any IP address of server.

User root //Line 48

Group root //Line 49

//Possess the authority of its group, always root. And the group exists in “/etc/group”.

#ServerAdmin root@localhost //Line 55

//The email address to which the server error alarm is sent to. It is annotated.

ErrorLog /dev/console //Line 62

//Error log file. If it does not start from “/xxx”, it starts from root path of the server. If the error log is not needed, modify the line into “ErroLog /dev/console” here. The boa printed information appears after system start-up is derived from “/dev/console”.

AccessLog /dev/null //Line 75

//Access log file. If it does not start from “/xxx”, it starts from root path of the server. If this log is not needed, modify the line into “AccessLog /dev/null” or annotate the line.

#UseLocaltime //Line 84

//Use local time or not. Use local time if this line is not annotated, otherwise use UTC time.

#VerboseCGILogs //Line 90

//Record CGI running information or not. Record if the line is not annotated, otherwise do not recored.

ServerName yellow //Line 95

//Server name.

#VirtualHost //Line 107

//Enable the virtual host or not. The device has many network interfaces, each interface is corresponding to a virtual Web server. This line is usually annotated.

DocumentRoot /web //Line 112

//The main directory preserving HTML file. If it does not start from “/xxx”, it starts from root path of server.

#UserDir public_html //Line 117

//When receiving a customer request, a new directory is added to the main directory.

DirectoryIndex index.html //Line 124

//The file name of HTML directory index. If no accessing directory is designated, this directory will be returned.

#DirectoryMaker /usr/lib/boa/boa_indexer //Line 131

//If there is no index file in HTML directory, and the user has not designated the access directory, “boa” would call the function to create an index file and return it to user. But this process is relatively a time consuming work. The user could also annotate this line and add index files to each HTML directory.

DirectoryCache /var/spool/boa/dircache //Line 140

//If “DirectoryIndex” does not exist, and “DirectoryMaker” has been annotated, the user needs to use the function carried by “boa” to create new directory index file in HTML directory (this directory needs to be readable and writable).

KeepAliveMax 1000 //Line 145

//The maximum request number of a connection supported by HTTP sustained action, Annotating this line or setting it to “0” will shut down the HTTP sustained action.

KeepAliveTimeout 10 //Line 149

//The waiting time interval between two requests of server in HTTP sustained action. The dimension is second. The connection is down if time is out.

MimeTypes /etc/mime.types //Line 156

//Designate the location of the file “mime.types”. If it dose not start from “/”, it starts from the root path of server. The user could choose to annotate this line and use “AddType” to designate clearly in the local file.

DefaultType text/plain //Line 161

//If the file has no extension name or the extension name is unkown, the default file type “MIME” is used.

CGIPath /bin:/usr/bin:/usr/sbin:/sbin //Line 165

//Provide the “PATH” environment parameters of CGI program.

#AddType application/x-httpd-cgi cgi //Line 174

//Associate the file extension name and the type “MIME”, the same as the funtion of “mime.types” file. The user could annotate this line if use “mime.types” file. Otherwise this line should not be annotated.

#Alias /doc /usr/doc //Line 189

//Designate the redirection path of the document.

ScriptAlias /cgi-bin/ /web/cgi-bin/ //Line 194

//Designate the actual path corresponding to the virtual path of CGI script. The CGI script is always placed in actual path, and the user input “site + virtual path + CGI script” to access. In this line, “/cgi-bin/” is the virtual path and “/web/cgi-bin/” is the actual path.

Save and exit the file.

Copy the file “mime.types” to the directory “etc/”. This file can always be found under “/etc” in PC.

#cp /etc/mime.types /opt/EmbedSky/root_nfs/etc

8.3.2 Configuring cgic library

Create the sub directory “cgi-bin/” under “web/” in file system:

#cd /opt/EmbedSky/root_nfs/web

#mkdir cgi-bin

Copy the transplanted cgic library and cgic test file to the directory “web/cgi-bin/”:

#cp /opt/EmbedSky/cgic205/capture /opt/EmbedSky/root_nfs/www/cgi-bin/

#cp /opt/EmbedSky/cgictest.cgi /opt/EmbedSky/root_nfs/www/cgi-bin/

8. 4 Testing

After finishing the previous operation, the user could boot the platform by NFS (the default IP address of platform is 192.168.1.6). After the platform start-up, run boa and start web server test.

8. 4. 1 Static web page test

We provide a testing web page in the file system. Input the following contents in web page explorer in PC.
http://192.168.1.6
The web page is open.

8. 4. 2 CGI script test

Use “helloweb.c” or “cgictest.cgi” for testing.
When using “cgictest.cgi”, enter the following contents in web page explorer in PC:
http://192.168.1.6/cgi-bin/cgictest.cgi
The web page is open.

Use “helloweb.c” for testing. The source code (under the directory “cgic205/”):

```
#include <stdio.h>
main()
{
    printf("Content-type: text/html\n\n");
    printf("<html>\n");
    printf("<head><title>CGI Output</title></head>\n");
    printf("<body>\n");
    printf("<hl>Hello, Web Server.</hl>\n");
    printf("<body>\n");
    printf("</html>\n");
    exit(0);
}
```

Compiling:

```
#arm-linux-gcc -o helloweb.cgi helloweb.c
#cp helloweb.cgi /opt/TQ/root_nfs/www/cgi-bin
```

Enter the following content in web page explorer in PC:
http://192.168.1.6/cgi-bin/helloweb.cgi
The web page is open

8. 5 Solving the error

8. 5. 1 Error 1

After “boa” starts up, the error “boa.c:266.icky Linux kernel bug! :No such file or directory” occurs. The solution: When configuring “boa.conf”, we set “User” and “Group” into “root”. Therefore, we need to annotate the contents from line 225 to line 227 in “boa.c”; Or we could choose to set “User” and “Group” into “nobody”, which needs the support of the file “etc/group” in file system. We select the first method.

8. 6 Some source codes

8. 6. 1 cgictest.cpp source code

```
//All variables need to be move to “MAIN” function when compiling.
// CGITEST.cpp : Defines the entry point for the console application.

//This program is used to test the upload character string information in WEB server.


#include "stdafx.h"                    //This line is annotated in LINUX.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
/*
                    void main()    //The return value in LINUX is integer. Add “return 0” or
                 “return 1” to the corresponding place.
{
 if(getenv("CONTENT-LENGTH"))
 {
  char *s = getenv("CONTENT-LENGTH");
  printf(s);
 }
 printf("Contenttype:text/html\n\n");
 printf("<html>\n");
 printf("<head><title>这是测试 POST 方法</title></head>\n");
 printf("<body><br>\n");
 printf("<h2> 这是测试 POST 方法</h2>\n");
 //printf(s);
```

```
 printf("<hr><p>\n");
 printf("<a><b> Go back to out put.html page </b></a>\n");
 printf("</body>\n");
 printf("</html>\n");
 fflush(stdout);
}
*/
/* convert hex string to int */
//Used for converting the format of Chinese character coding.
int htoi(char *s)
{
 char *digits="0123456789ABCDEF";
 if (islower(s[0])) s[0]=toupper(s[0]);
 if (islower(s[1])) s[1]=toupper(s[1]);
 return 16 * (strchr(digits, s[0]) -strchr (digits,'0'))+(strchr(digits,s[1])-strchr(digits,'0'));
}

void main()
{
 printf ("Contenttype: text/plain\n\n");
 printf("<html>\n");
printf("<head><title>这是测试 POST 方法</title></head>\n");
 printf("<body bgcolor=#008080 text=#FFFFFF><br>\n");
// printf("<p align=center><img border=0 src=http:
//127.0.0.1:8080/winter.gif width=750 height=120></p>");
 printf("<p align=center><img border=0 src=/winter.gif width= 700 height=120></p>");
 printf("<hr noshade color=#FF0000>");
 printf("<h2>  这是测试 POST 方法</h2>\n");
 printf("<hr noshade color=#FF0000>");
/*************************************************************************/
//Place the gotten value in "nValue"
 int i,n;
 char c;
 int nSum = 1;                    //The number of variables
 char nStr[1000];                 //Used for storing the upload character string, 1000 is the
maximum number.
 memset(nStr,0,1000);             //Clear the 10 variables.

// char nCurrentValue[200];        //The current gotten value.
// char nValueName[10][50];        //Variable name
// memset(nValueName,0,500);       //Clear the name of 10 variables.
 char nValue[10][100];            //10 varibles is the maximum number, maximum 100 characters
for each variable.
```

```
memset(nValue,0,1000);                    //Clear the 10 variables.
int nIndex = 0;                           //Current variable index.
int nPosion = 0;                          //The serial number of the variable currently operated.
int iseq=0;                               //The start flag of each variable.
n=0;
if(getenv("CONTENT_LENGTH") == NULL)
{
 return;                                  //No "CONTENT_LENGTH" environment parameter exists in
web server environment.

}
n=atoi(getenv("CONTENT_LENGTH"));   //Convert the length of character string into integer type value
printf("数据长度%d<br>",n);
for (i=0;i<n;i++)
{

 c=getchar();                             //Get a character from standard input.
 nStr[i]=c;

//The following contents are mainly about the URL coding and decoding.
 switch (c)
 {
  case '&':
nSum += 1 ;                           //The number of variables.
nIndex += 1;                          //The index number of variable.
nPosion = 0;                          //Clear the character position.
//c=' ';
iseq = 0;                             //Clear the start flag of variable.
break;
  case '+':                               //Space key conversion.
//c=' ';
if(iseq == 1)
{
   nValue[nIndex][nPosion] = ' ';
   nPosion ++;
   }
  break;
  case '%':                               //No number or letter, for example, Chinese character coding
conversion.
{
   char s[3];
   s[0] = getchar();
   s[1] = getchar();
```

```
    s[2] = 0;
    c = htoi(s);
    i += 2;                              //Should be in pairs.
    if(iseq == 1)
    {
     nValue[nIndex][nPosion] = c;
     nPosion ++;
    }
}
  break;
 case '=':                               //The start of variable.
//c = '=';
  iseq = 1;
  nPosion = 0;                           //The first character of the variable currently being operated.
  break;
 default:                                //Other character.
{
    if(iseq)
    {
     nValue[nIndex][nPosion] = c;
     nPosion ++;
    }
}
  break;
 }
//  putchar(c);
 fflush(stdout);
 }

/*****************************************************************************/
 nStr[n] = '\n';
 printf("<br>");
 printf("变量个数  =  %d",nSum);
 printf("<br>");
 printf("nIndex 数  =  %d",nIndex);
 printf("<br>");
 printf("nPosion 数  =  %d",nPosion);
 printf("<br>");

 for(i=0; i<nSum; i++)
 {
  printf("第%d 个上传的值:%s",i+1,&nValue[i][0]);
  printf("<br>");
```

```
}
printf(nStr);                                   //Display the POST upload character string

printf("<hr noshade color=#0000FF>");
/*****************************************************************************/
printf("<br>");
printf("调用该 CGI 程序的网页的 URL：%s",getenv("HTTP_REFERER"));

printf("<br>");
printf("调用该 CGI 程序的 Web 浏览器的机器名和域名：%s",getenv("REMOTE_HOST"));
printf("<br>");
printf("IP 地址和主机名：%s",getenv("REMOTE_ADDR"));
printf("<br>");
printf("服务器的 IP 或名字：%s",getenv("SERVER_NAME"));
printf("<br>");
printf("主机的端口号：%s",getenv("SERVER_PORT"));
printf("<br>");
printf("服务器软件的名字：%s",getenv("SERVER_SOFTWARE"));
printf("<br>");
printf("用户和组名：%s",getenv("REMOTE_USER"));

printf("<br>");
printf("Web 服务器传递数据给 CGI 程序时所采用的方法%s",getenv("REQUEST_METHOD"));
printf("<br>");
printf("发送给服务器的完整 URL 请求：%s",getenv("REQUEST_LINE"));

printf("<br>");
printf("该 CGI 程序的名称：%s",getenv("SCRIPT_NAME"));

printf("<br>");
printf("QUERY-STRING：%s",getenv("QUERY_STRING"));
printf("<br>");


/*****************************************************************************/
printf("<hr noshade color=#00FF00>");
printf("<a><b> 数据上传测试！ </b></a>\n");
printf("</body>\n");
printf("</html>\n");
fflush(stdout);

}
void GetOnePostChar()
```

```
{

}
```

8. 6. 2 Source code of home page

```
<html>
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>CGI 数据上传测试</title>
</head>
<body background="bg.jpg" style="background-attachment: fixed">
<p align="center"><img border="0" src="water.gif" width= 95%   height="120"></p>
<hr noshade color="#FF0000">
<form method="POST" action="cgi/CGITEST.CGI" name="form">

 <table border="4" width="100%" id="table1" bordercolor="#0000FF" align="left" style="border-collapse: collapse" height="197">
  <tr>
   <td width="187" height="55"><blink>
   <font face="华文行楷" size="5" color="#0000FF">姓名：</font></blink>
   </td>
   <td width="84" height="55">
   <p align="center">
 <input name="NAME" size="10" value="Yellow" style="float: left"></td>
   <td rowspan="3" width="306">
   <p align="center"> </p>
   <p align="center">   </p>
   <p align="center">IP 地址：   <input type="text" name="T1" size="20" value="192.168.1.6"></p>
   <p align="center">子网掩码：<input type="text" name="T2" size="20" value="255.255.255.0"></p>
   <p align="center">   </p>
   <p align="center">网关地址：<input type="text" name="T3" size="20" value="192.168.1.2"></p>
   <p align="center">组播地址：<input type="text" name="T4" size="20" value="234.5.6.7"></p>
   <p align="center">   </td>
   <td rowspan="3">
   <p align="center">用户名：
   <input type="text" name="T5" size="20" value="administrator"></p>
   <p>   </p>
   <p align="center">口令密码：<input type="password" name="T6" size="20" value="123456"></td>
  </tr>
  <tr>
```

```
 <td width="187"><font face="华文行楷" size="5" color="#0000FF">性别：</font></td>
 <td width="84">
 <p align="left">
<select size="1" name="SEX" style="border-style: solid; border-width: 1px; padding-left: 4px;
padding-right: 4px; padding-top: 1px; padding-bottom: 1px" >
<option selected value="男">男</option>
<option>女</option>
</select></td>
 </tr>
 <tr>
 <td width="187" height="109">
 <font face="华文行楷" size="5" color="#0000FF">年龄：</font></td>
 <td width="84" height="109">
 <p align="center">
<input name="AGE" size="10" value="38" style="float: left"></td>
 </tr>
</table>
<p align="center">　</p>
<p align="center">　</p>
<p align="left" ><font face="华文行楷" size="5" color="#FFFF00">　</font>
</p>
<p>
</p>
<p align="center">
　</p>
<p align="center">
　</p>
<p align="center">
　</p>
<p align="center">
<button name="B2" style="width: 60px; height: 40px" type="submit" value="SEND INFORMATION">
<p>上传数据</button></p>
</form>

<hr noshade color="#FF0000">

<p align="center"><img border="0" src="03y_15.gif" width="290" height="429"></p>
</body>
</html>
```

8. 7 Environment variable

8. 7. 1 The environment variable relevant to the server

GATEWAY_INTERFACE	The CGI version obeyed by the server
SERVER_NAME	The IP address or name of the server
SERVER_PORT	The port number of the host
SERVER_SOFTWARE	The name of the server software

8. 7. 2 The environment variable relevant to the customer

The customer environment is probably unknown to the server, but it is important because it is relevant to the explorer and so on.

ACCEPT	List the response type that could be accepted by request.
ACCEPT_ENCODING	List the coding type that could be supported by the customer
ACCEPT_LANGUAGE	Indicate the ISO code that could be accepted by customer
AUTORIZATION	Indicate the qualified customers
FORM	List the EMAIL address of customers
IF_MODIFIED_SINGCE	Return value only when adopting "get" request method and file is older than the designated date
PRAGMA	Set the probably used proxy
REFFERER	Indicate the URL of the document currently connected to
USER_AGENT	Indicate the software used by the customer

8. 7. 3 The environment variable relevant to the request

The CGI application should be aware of all information, because requests could not be same. The information relevant to requests contains important elements, like user calling information, request type and the percentage of transmitted information. The following contents introduce detailed the information, expecially the 3 variables.

REQUEST_METHOD
QUERY_STRING
CONTENT_LENGTH

These 3 variables are very important, which represent the process of transmitting data to CGI program. The user could make good use of them, such as, get to know the opponents are calling your function, the use has registered or not, and connect to your CGI program to set path information to be contained in request. And there is no need to guess in which page of the server you are at.

AUTH_TYPE	The verification mode of the server
CONTENT_FILE	The data file containing CGI program
CONTENT_LENGTH POST	The number of bytes sent to standard input (STDIN) in request

CONTENT_TYPE	The type of data that is sent
PATH_INFO CGI	The added path of the program
PATH_TRANSLATED PATH_INFO	The corresponding absolute path
QUERY_STRING	The part after "?" of URL sent to CGI program
REMOTE_ADDR	The IP or host name of the terminal user
REMOTE_USER	The group name of user if the user is legal
REQUEST_LINE	The complete URL request sent to server
REQUEST_METHOD	The data sending method as part of the HTTP request, like "get"
SCRPT_NAME	The name of running script

Chapter 9 Embeded database transplantation (SQLite)

The basic objectives of database are data storage, search and so on. Besides the function of search, add, delete, the traditional database also provides many advanced features, like trigger, storage process, data backup, information recover and so on. However, in most situations, the frequently used functions are not the advanced ones, but the basic ones. And the traditional database is always too huge in size, especially for the embedded system. Therefore, the small-sized embedded database begins to win more attention.

Embeded database possesses the general features of database. The difference between embedded database and traditional database: Embeded database is directly driven by program, but the tranditional database is driven by engine response; Embeded database is always small in size, which can be conveniently transplanted to handheld devices.

9. 1 Introduction of SQLite database

SQLite database is a kind of embedded database which is oriented to a simplified usage. This database avoids the complex features of the traditional enterprise database, and only keeps the basic database functions.

The function and performance of SQLite are both excellent, although it is some kind of a simplified database. SQLite supports the following features:

- Support ACID affair (ACID is the abbreviation of Automic, Consisten, Isolated and Durable)
- No managerial configuration is needed
- Support SQL92 norm
- Data are placed in different files. The maximum size of file supported by SQLite reaches 2TB
- Database could be shared among different devices with different supporting bytes
- Small sized
- Small system overhead and high searching efficiency
- Easy-used API interface
- Could be bound with many languages, like “Tcl”, “Python”, “C/C++”, “Java”, “Ruby”, “Lua”, “Perl” and “PHP”.
- No external support is needed
- The code is well annotated
- Over 95% of the code have been tested
- Open source

SQLite has the advantages of powerful function, simple interface, high speed, small size and so on. And it is especially suitable for embedded system and secondary development. Here we set a chapter to introduce the SQLite transplantation in ARM-Linux and the brief test. Hoping it would be helpful.

9. 2 Transplanting SQLite database

9. 2. 1 Get SQLite source code

Access the web site http://sqlite.org/download.html and get the latest SQLite source code.

We provide the SQLite source code of version 3.5.9, which is namely the file “SQLite.tar.bz2” under the directory “\Linux\linux-2.6.13\” in TQ2440 CD-ROM.

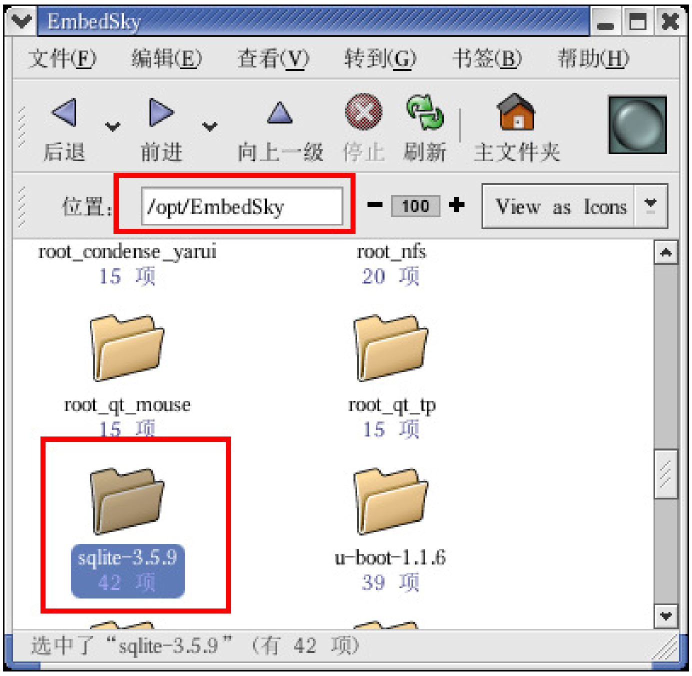
The source code of SQLite contains a simple test program, under the directory “sqlite_test” in source code package.

9. 2. 2 Transplanting SQLite

Step1: Decompress the source code package. Copy the package to the directory “/opt/EmbedSky” in Linux in PC, and execute the following command to decompress it:

#tar xvfj SQLite.tar.bz2 -C /

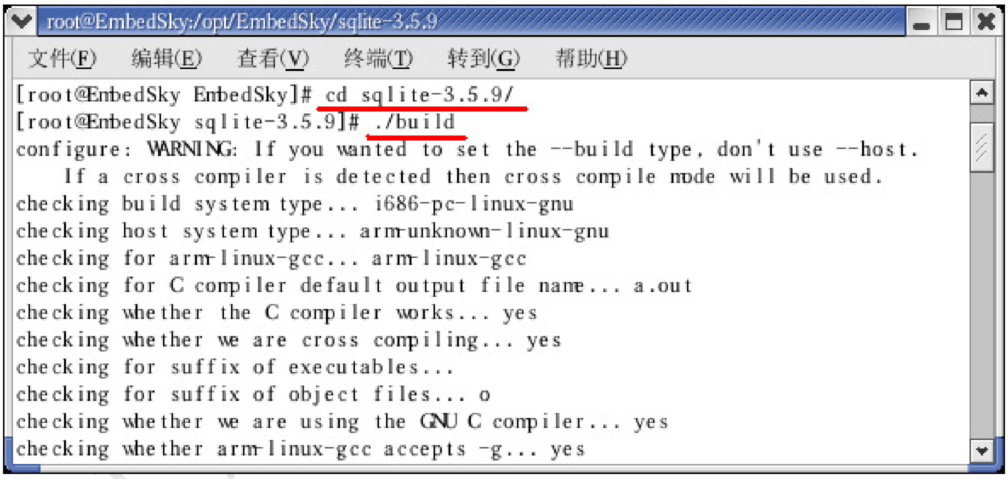
After decompression, the directory “sqlite-3.5.9” is auto-created under “/opt/EmbedSky”:



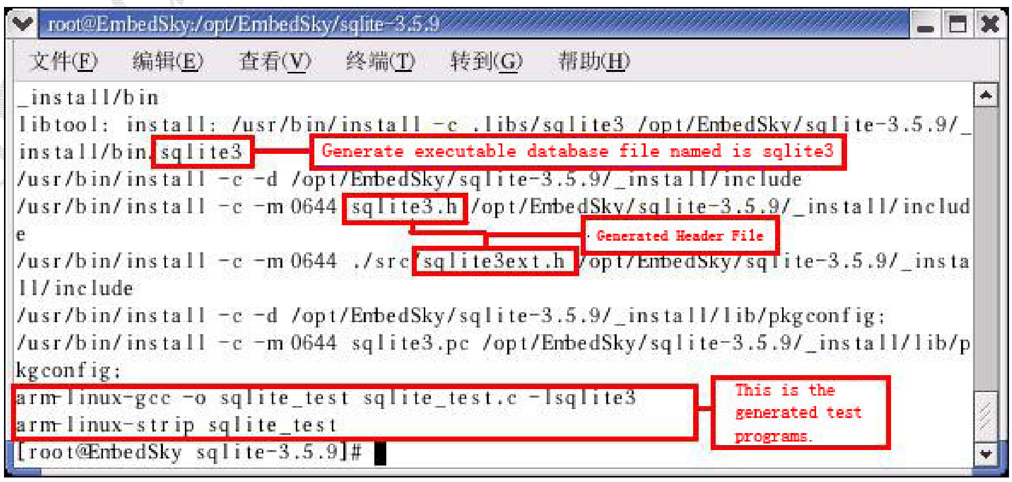
Step2, configuring SQLite. Get into the directory “sqlite-3.5.9” and use the compiling script file “build” provided by us to compile SQLite. Executing the following command:

#build

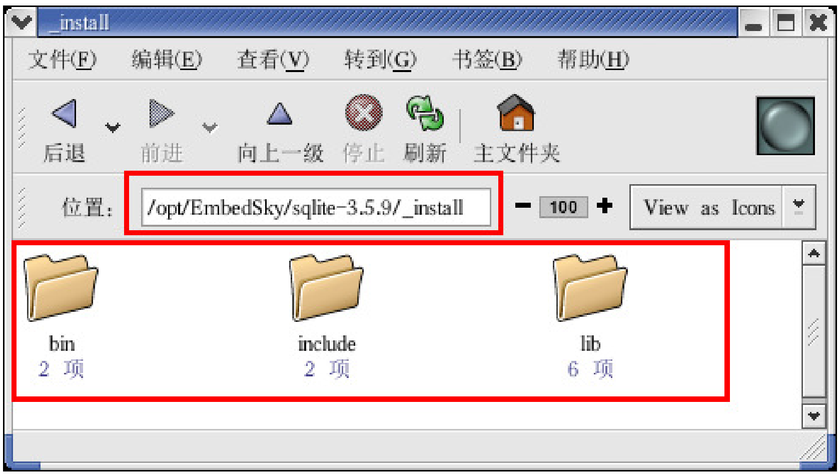
As shown in the following diagram:



Compiling is complete:



Installation automatically starts after compiling. The database program “sqlite3” and the test program “sqlite_test” are under the directory “bin” in “sqlite-3.5.9/_install”; The head file of database is under the directory “include” in “sqlite-3.5.9/_install”; Library file is under the directory “lib” in “sqlite-3.5.9/_install”:



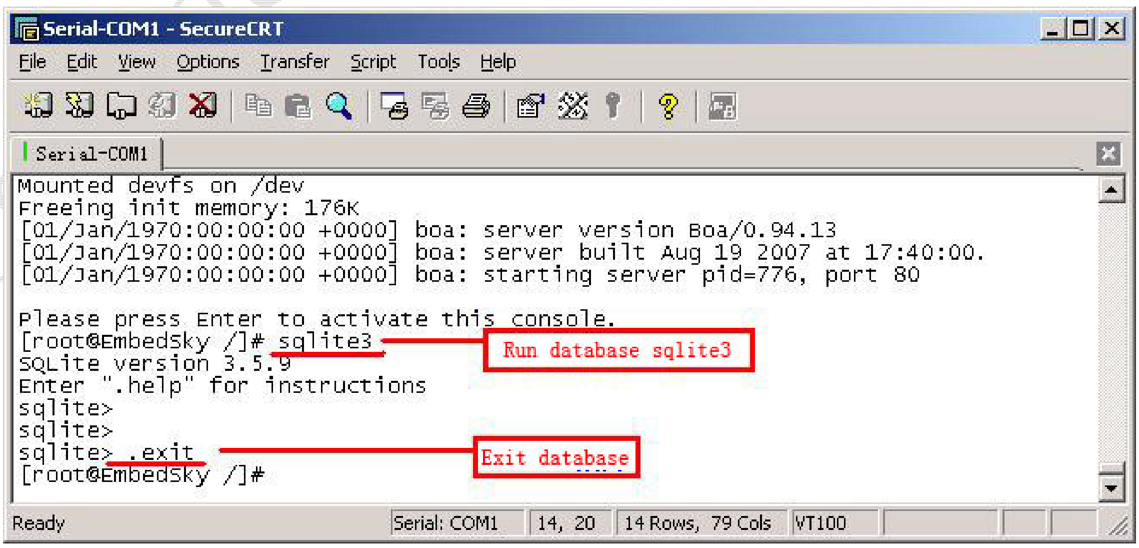
9. 2. 3 Using SQLite

Copy the files “sqlite3” and “sqlite_test” gained after compiling to the directory “/bin” of file system, and copy the library files under the directory “lib” to “/lib” in file system.

Copy the files under the diretory “include” to “/include” of cross-compiler. And copy the files under the directory “lib” to “/lib” of cross-compiler. Therefore, the cross-compiler is able to support “sqlite3” database (the corresponding files has been added to cross-compiler by us).

9. 3 Testing SQLite database

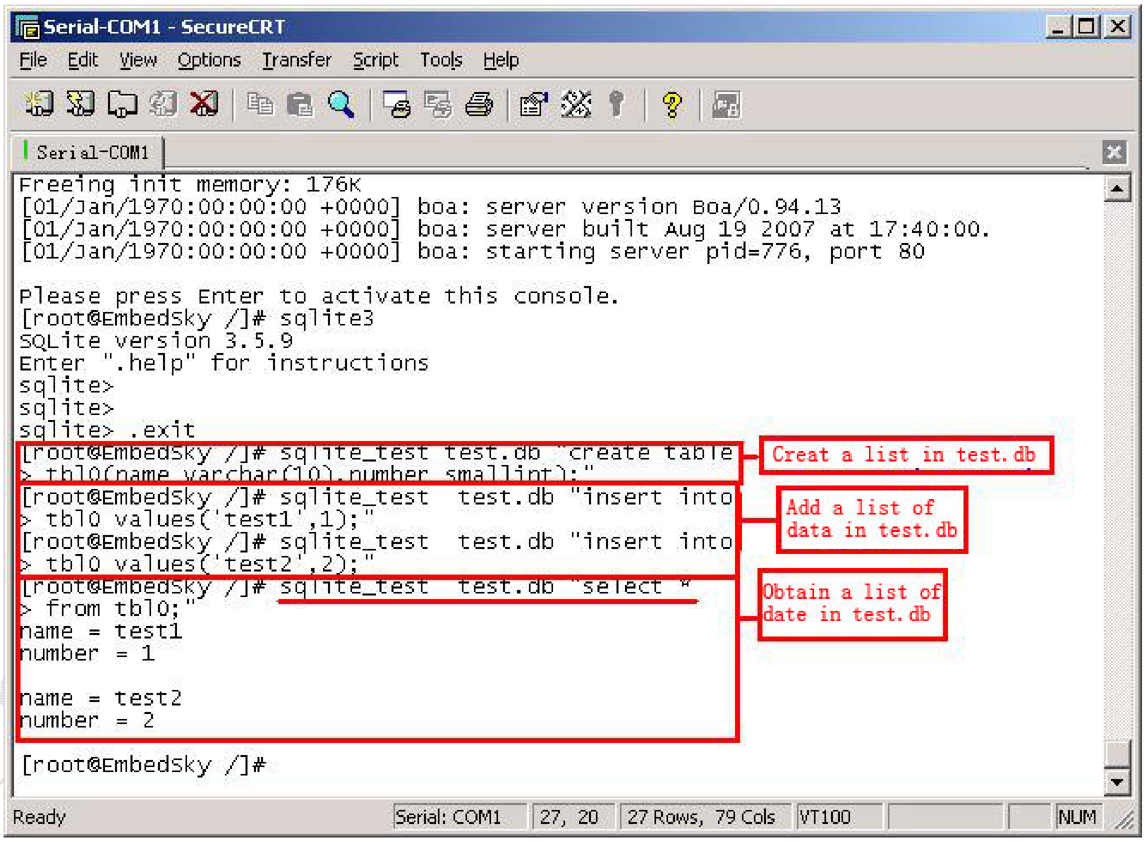
Running “sqlite” database solely:



Use “sqlite_test” program to test the database. The operation steps in platform are shown as follows:

```
[root@EmbedSky /]# sqlite_test test.db "create table
> tbl0(name varchar(10),number smallint);"
[root@EmbedSky /]# sqlite_test   test.db "insert into
> tbl0 values('test1',1);"
[root@EmbedSky /]# sqlite_test   test.db "insert into
> tbl0 values('test2',2);"
[root@EmbedSky /]# sqlite_test   test.db "select *
> from tbl0;"
name = test1
number = 1

name = test2
number = 2

[root@EmbedSky /]#
```

The operating states:


Serial-COM1 - SecureCRT
File Edit View Options Transfer Script Tools Help
Serial-COM1
Freeing init memory: 176K
[01/Jan/1970:00:00:00 +0000] boa: server version Boa/0.94.13
[01/Jan/1970:00:00:00 +0000] boa: server built Aug 19 2007 at 17:40:00.
[01/Jan/1970:00:00:00 +0000] boa: starting server pid=776, port 80
Please press Enter to activate this console.
[root@EmbedSky /]# sqlite3
SQLite version 3.5.9
Enter ".help" for instructions
sqlite>
sqlite>
sqlite> .exit
[root@EmbedSky /]# sqlite_test test.db "create table
> tbl0(name varchar(10),number smallint);"
[root@EmbedSky /]# sqlite_test test.db "insert into
> tbl0 values('test1',1);"
[root@EmbedSky /]# sqlite_test test.db "insert into
> tbl0 values('test2',2);"
[root@EmbedSky /]# sqlite_test test.db "select *
> from tbl0;"
name = test1
number = 1
name = test2
number = 2
[root@EmbedSky /]#
Creat a list in test.db
Add a list of data in test.db
Obtain a list of date in test.db
Ready
Serial: COM1
27, 20
27 Rows, 79 Cols
VT100
NUM