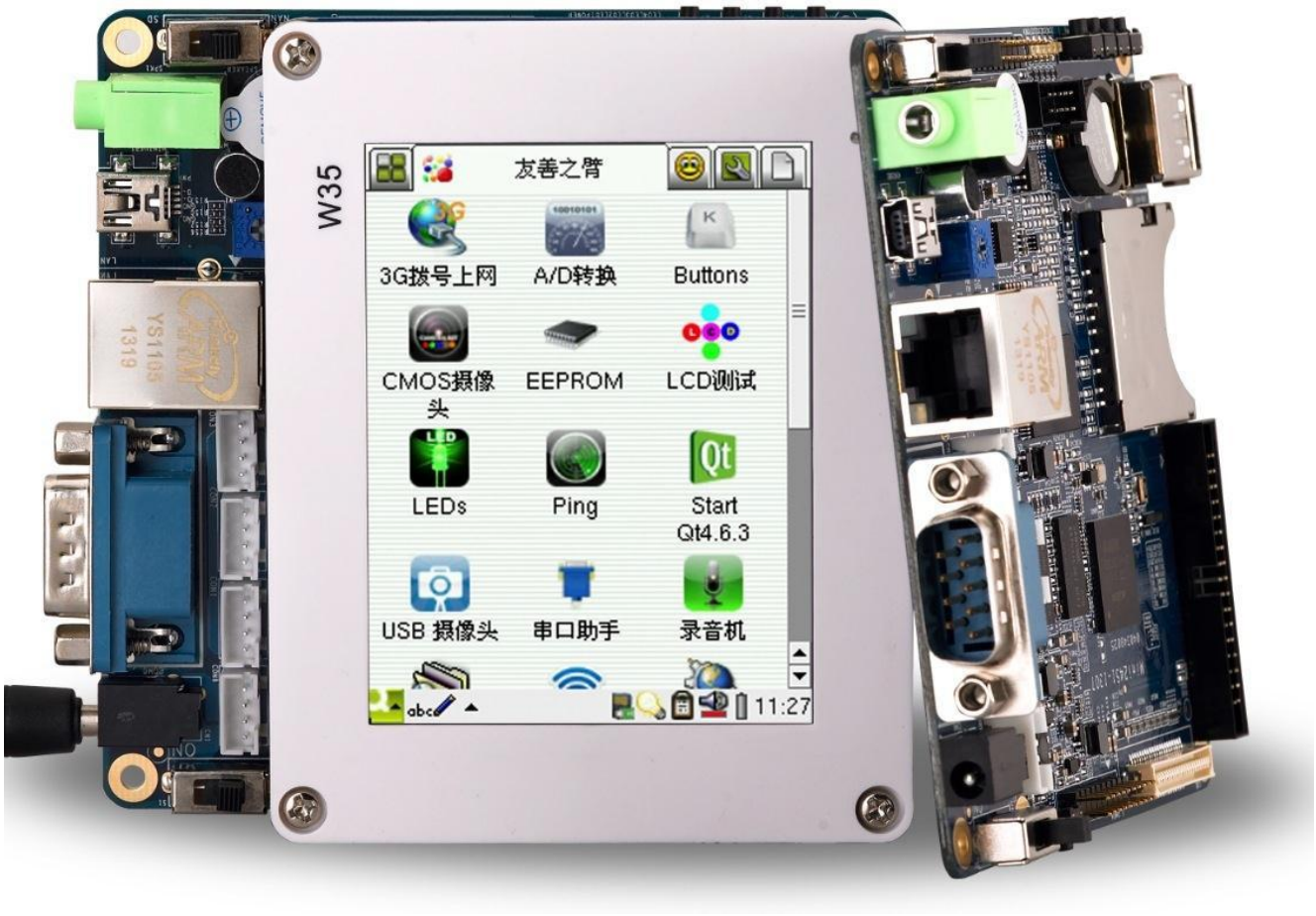
Mini2451 User's Manual



REVISION	ORIGINATOR	SCR	REV DATE
0.1.0	Guangzhou FriendlyARM Co., Ltd		Spetember 26, 2013
Guangzhou FriendlyARM Computer Tech Co., Ltd Confidential: This document and information contained in it shall not be reproduced by, used by, or disclosed to others except as expressly authorized in writing by Guangzhou FriendlyARM Co., Ltd.			**Guangzhou FriendlyARM Computer Tech Co., Ltd Guangzhou, China**

Address: Room 1705,Block A1, Longyuan Plaza, Longkouxi Road, Guangzhou, China, 510640 Website: http://www.arm9.net
Sales: +86-20-85201025 Tech Support: +86-13719442657 Fax: +86-20-85261505
Email for Business and Cooperation: capbily@163.com Email for Tech Support: dev_friendlyarm@163.com

COPYRIGHT STATEMENT

Index

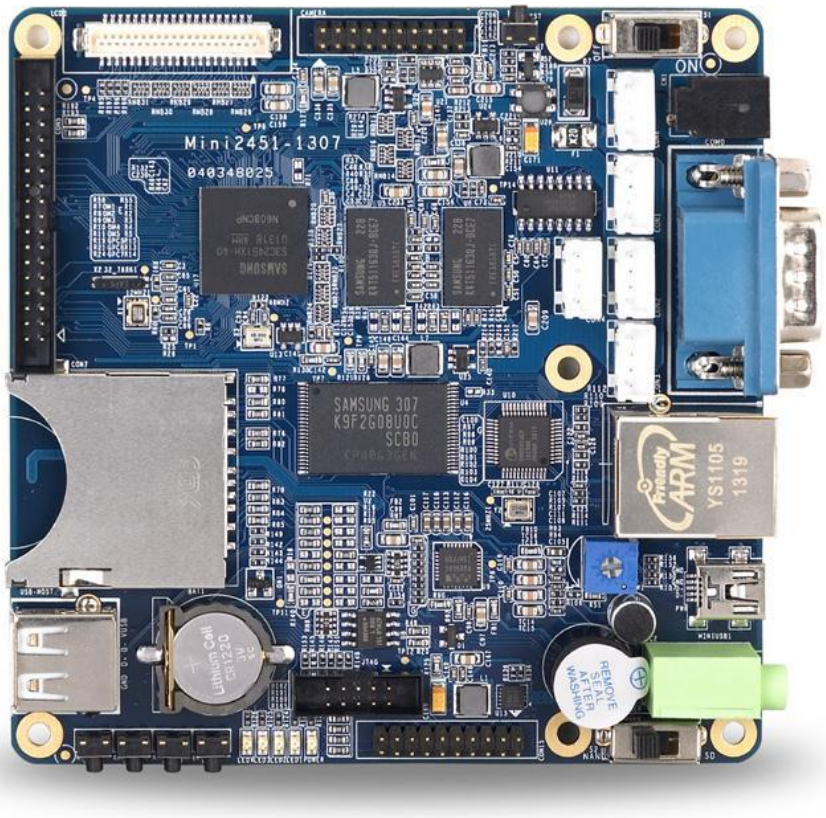
1 Introduction

The Mini2451 development board is an ARM9 embedded processing board that uses the Samsung S3C2451 System On Chip (SOC). Its highest frequency is 522MHz. As a successor of the Mini2440 its ports, interfaces and layout is almost 99% compatible with the Mini2440. In addition, compared to the Mini2440 the Mini2451 has a better power system and overall design therefore it can avoid strong electrical interference and extreme temperature fluctuations. These features make it fit the most needs of outdoor applications

Compared to the Mini2440 the Mini2451 runs much faster and can work with larger RAMs. It has four serial ports. Its USB supports USB2.0. The 2451 supports SD card booting and therefore we developed a Superboot-2451 which enables installing OS via an SD card.

1.1 Mini2451 Board

Here is a Mini2451's photo.



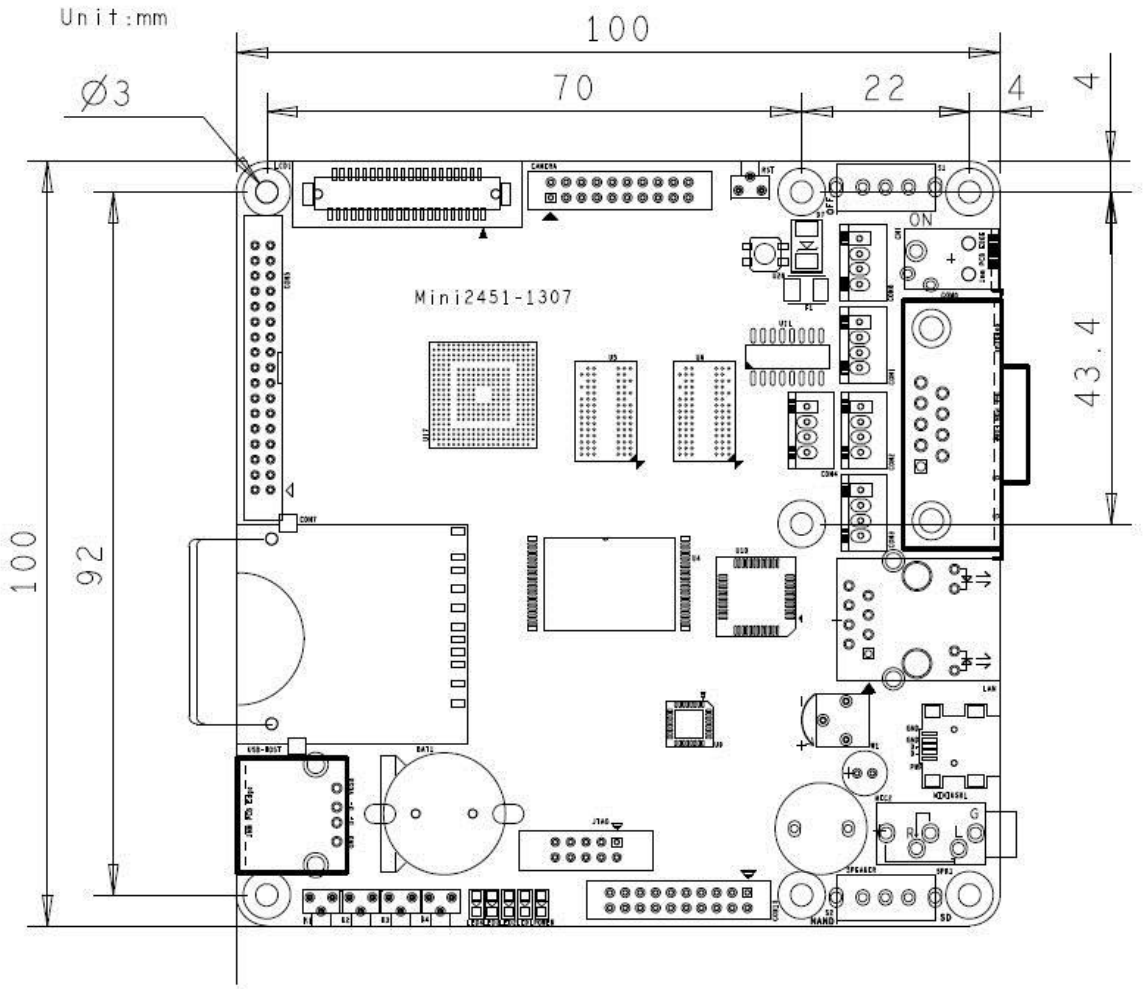
1.1.1 Mini2451 Hardware Feature

CPU	● Samsung S3C2451, based on ARM926EJ, 400MHz , max 522MHz
RAM	● 128MB DDR2 RAM ● 133 M Hz
FLASH	● SLC NAND Flash: 256MB, optional 256MB/1GB
LCD	● LCD interface: 41Pin, 1.0 mm spacing, compatible with Mini2440/Mini6410/Mini210S LCD, supports one wire precise touching ● LCDs supported from 3.5” to 12.1” , resolution up to 1024 x 768
Network	● 10/100M Ethernet interface(RJ45) using DM9000AEP

Standard Configuration	● 1 x DB9 RS232 serial port ● 1 x miniUSB 2.0 ● 1 x 3.5mm stereotype audio output ● 1 x integrated microphone ● 1 x one speaker port which can drive an 8Ω 1W speaker ● 1 x USB Host 2.0 ● 1 x standard SD card socket ● 1 x 5V power input (DC-23B)
On Board Hardware Resource	● 1 x I2C-EEPROM (256byte) for I2C bus test ● 4 x LED (Green) ● 4 x Interrupt Style Push Button on module ● 1 x adjustable resistor for ADC testing ● 1 x PWM buzzer ● 1 x backup battery for on board real time clock
External Resource	● 4 x TTL socket, 2.0 mm spacing, three wire serial port ● 1 x JTAG: 10 pin, 2.0mm spacing ● 1 x SDIO: 20 pin, 2.0mm spacing, includes 1 x SPI, 1 x IIC and 1 x serial port, for SD WiFi ● 1 x CMOS: 20pin, 2.0mm spacing, can work with CCD cameras ● 1 x GPIO (including SDIO): 34 pin, 2.0mm spacing, includes AD inputs, interrupts, I2C, SPI, PWM and 5V&3.3V power pin
Power	● 3.7V to 6V (support sleep mode)
PCB Dimension	● Dimension: 100 x 100 (mm)

Software	● Superboot-2451 ● Linux 3.6 + Qtopia 2.2.0/Qt 4.4.3/Qt 4.8.5 ● WindowsCE 6.0 ● uCos2

1.1.2 Schematics



1.2 Mini2451 Software Feature

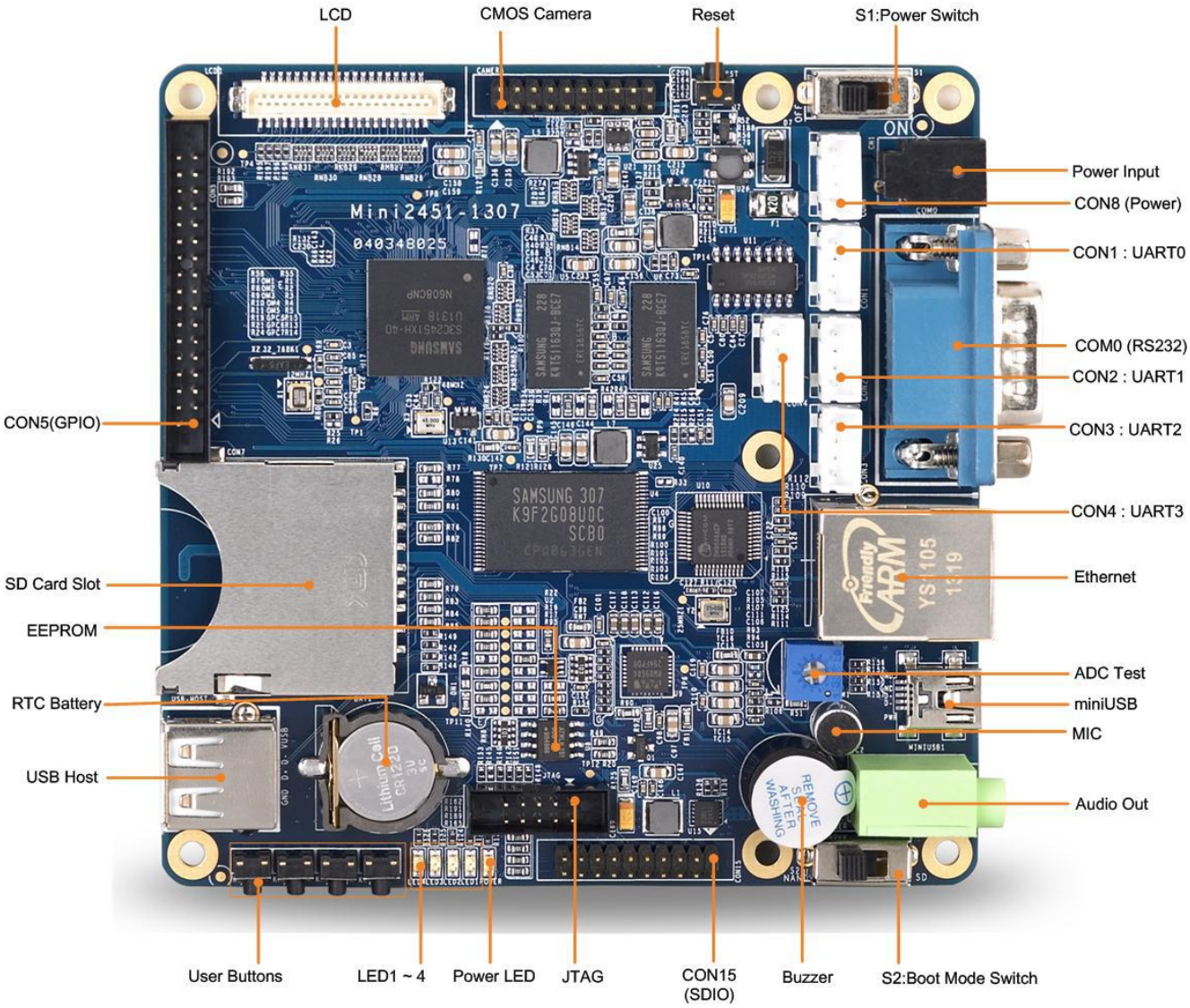
1.2.1 Linux Feature

Cross-compiler	arm-linux-gcc-4.4.3	Same as Mini2440, it supports hard floating point arithmetic
Superboot-2451	It supports SD card burning and can install (Linux/WinCE/uCos2) systems.	
	It supports bin file, YAFFS2/EXT3/EXT4 etc.	
	It supports Nand Flash burning.	
Linux kernel	Kernel version: Linux-3.6	Complete BSP
	It supports YAFFS2/CRAMFS/NFS/UBIFS/FAT32.	Open source
	Watchdog	Open source
	RTC driver	Open source
	LED driver	Open source
	User button driver	Open source
	SPI driver	Open source
	I2C-EEPROM driver	Open source
	PWM buzzer driver	Open source
	ADC driver (channel: AIN0)	Open source
	Driver for ARM touch screen	Open source
	One wire precise touching driver	Open source
	LCD back light driver: it allows users to adjust the board's backlight up to 127 levels	Open source
	LCD driver(4.3", 5", 7", 8" and 10")	Open source
	USB Host driver: it supports flash drives, blue	Open source

	tooth and so on.	
	USB Device driver	Open source
	SD card driver	Open source
	Serial port driver	Open source
	SD WiFi driver	N/A
	USB WiFi driver: it comes with the kernel but can only drive limited types	Open source
	USB WiFi driver: it supports more types	
	Audio driver(WM8960: it supports audio recording and playing, ALSA API and type D amplifier）	
	Ethernet driver(DM9000AEP)	Open source
	CMOS driver	N/A
	2D accelerator	Open source
	USB to Serial driver	Open source
	3G driver	Open source
GUI System	Version: Qtopia 2.2.0	Open source. It includes code for both X86 and ARM
	Qt/Embedded 4.8.5	Open source. It includes code for ARM
Application	3G Dial-up utility	It supports WCDMA, CDMA2000, TD-SCDMA and auto dial-up.
	GPRS messaging	It supports serial port, USB or GPRS Modem messaging and group messaging.
	ADC test utility	
	LED test utility	
	Button test utility	
	I2C test	EEPROM read and write
	LCD test	It supports manual and auto modes

	Ping	
	USB camera	
	CMOS camera	
	Audio recording	ALSA
	Web browser	
	Watchdog	
	Network configuration utility	It can be used to set up IP, DNS, gateway etc
	Backlight control	
	Language setting	It includes three options: Chinese, English and Japanese
	Touch pen test utility	
	MMC/SD card mount/unmount utility	
	Qt-4.8.5 switcher	It can be used to switch between Qtopia 2.2.0 and Qt 4.8.5

1.3 Interface and Port



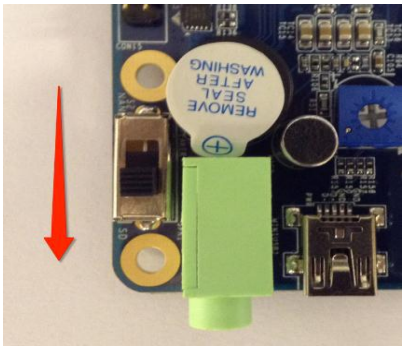
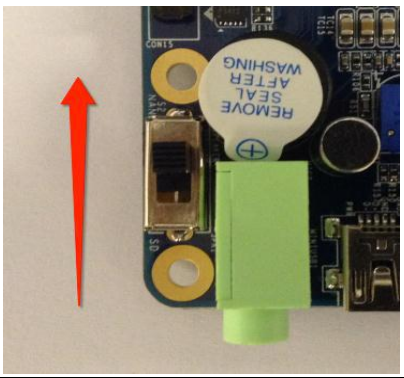
2 Get Started

By default, all our boards have been preinstalled with Linux therefore you can just power on and play.

2.1 System Setup and Configurations

2.1.1 Boot Option

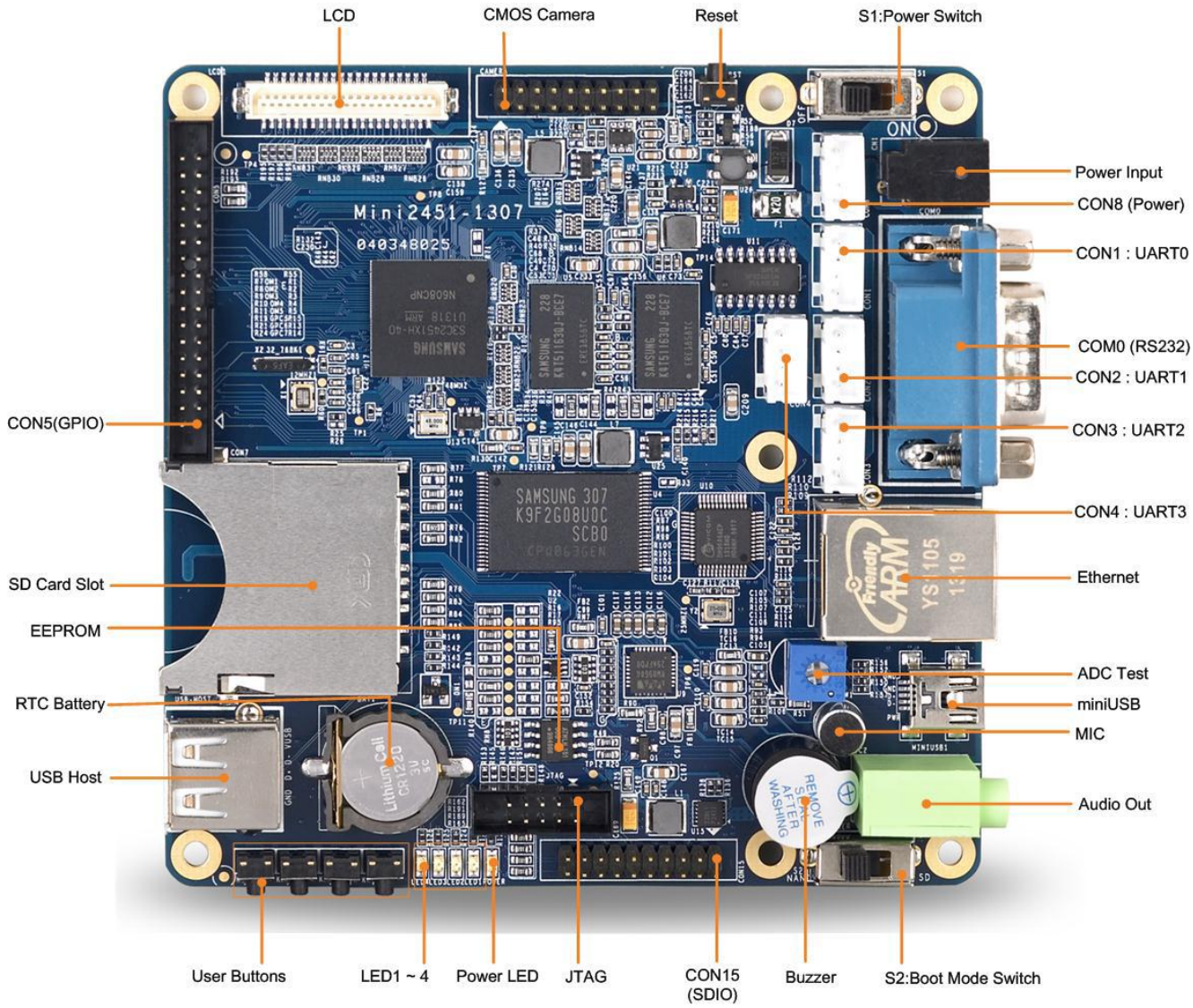
The Mini2451 supports booting from either SD card or Nand Flash. Users can switch between by toggling the S2 switch:

Screenshot	Operation	Comment
	Toggle S2 to "SD" and the board will boot from your SD card	This is for booting from SD card or system burning
	Toggle S2 to "NAND" and the board will boot from the NAND Flash	Default and Standard booting.

Usually, S2 is switched to "Nand" unless users need to boot from an SD card or reflash the system.

2.1.2 Interface and Port

The Mini2451 has the following interfaces, ports and sockets:



Please follow the steps below to hook up the board:

- Connect the Mini2451 board's serial port0 (Debug Serial Port) to a PC's serial port with the shipped crossover serial cable (blue one) in the package
- Connect the Mini2451 board's Ethernet to a PC with the shipped crossover cable

(this step can be skipped if you don't need to connect to the internet)

- Connect the shipped 5V power supply adapter to the 5V power supply interface on the board (do it with care to prevent damaging the interface)
- Connect a headphone or speaker to the audio input(green) on the board
- Connect an LCD touch screen (if the user has one) to the LCD interface on the board following the data bus' arrow

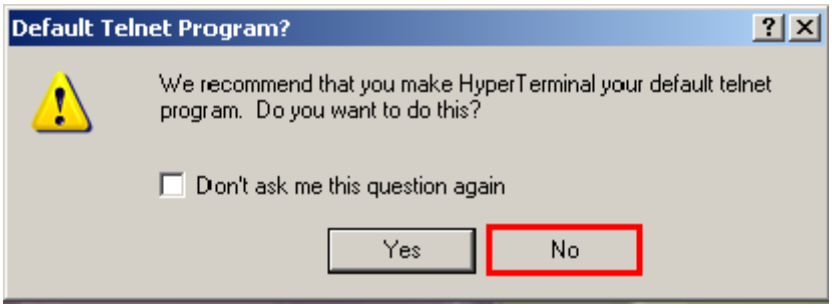
2.1.3 Setting up HyperTerminal

Note: some users attempt to expand the board's serial ports by using a USB to serial port cable. Sometimes this operation would cause error codes. This might be an indication that the cable doesn't work. Most of our agents have these conversion cables for sale. Users can contact them. In addition we strongly recommend users to use our shipped crossover serial cable. Other serial cables might not work properly.

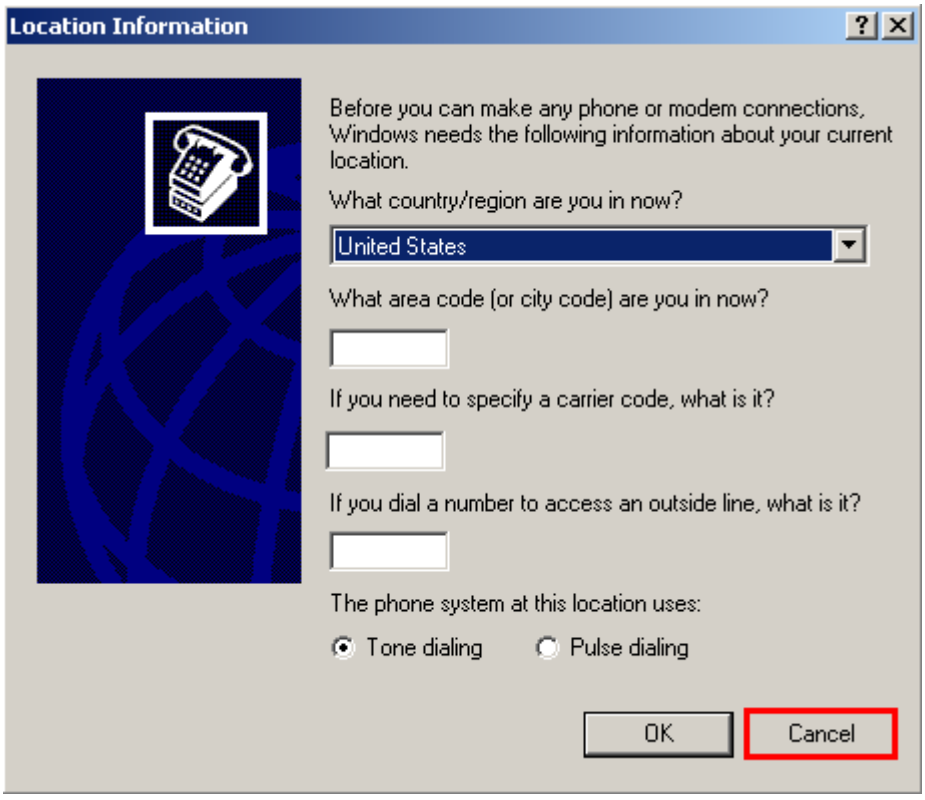
To connect the Mini2451 board to a host PC via a serial cable, you should use a simulation terminal. There are many tools available. A most widely used one is the MS-Windows' super terminal. In Windows9x, you need to install it by checking that option during installation. Windows2000 and later

A common Linux desktop version has a similar terminal too and it is minicom. It is a command line utility which may not be easy for beginners. Interested users can search the internet for more resources.

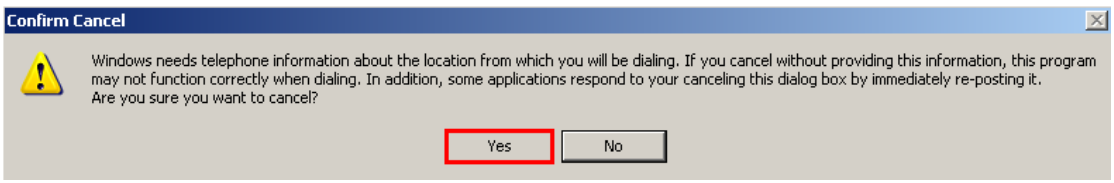
We take WindowsXP's super terminal for instance. You can find it by going to "Start->Programs->Accessories->Communications". After it starts the following dialog will pop up, please click on the "No" button

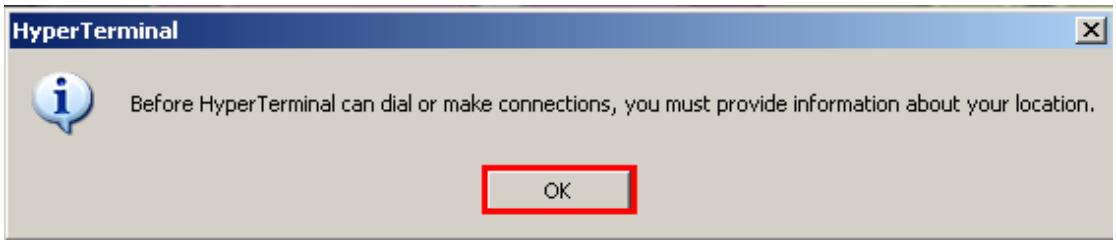


Click on the "Cancel" button on the following dialog



Click on the "Yes" button and the "OK" button to the next step



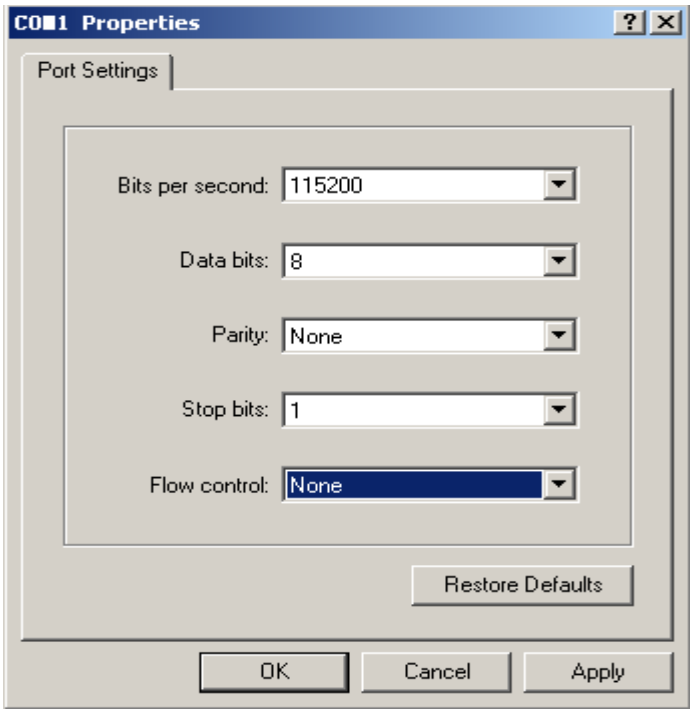


A popup window will require you to name this connection. In this example we typed "ttyS0". Windows does not accept names like "COM1" that have already been used by the system.

After naming this connection another window will require you to select a serial port that will be used to connect the Mini2451 board. Here we selected COM1:

Lastly, also the most important step is to set up the port properties. Note: you

must select "No" in the data flow control field otherwise you will only be able to see outputs. In addition the bits per second should be set to 115200.



After setting up all properties, turn on the board's power supply, if the connection gets set properly, you will see a bootloader startup interface. If everything runs well please save this connection.

2.2 Burn Superboot to SD Card

In order to boot from an SD card, you need to burn BIOS to it. FriendlyARM offers a flashing utility: SD-Flasher.exe which can burn our Bootloader (Superboot2451.bin) to an SD card.

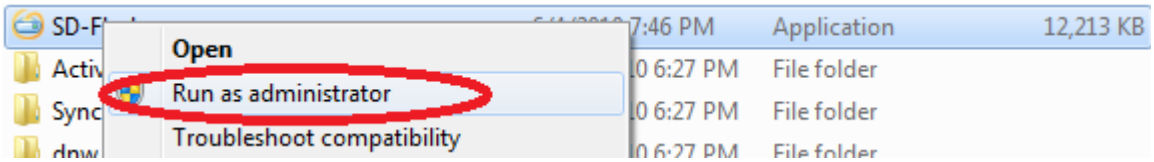
2.2.1 Burn Superboot to SD Card

We tested the following steps on Windows7

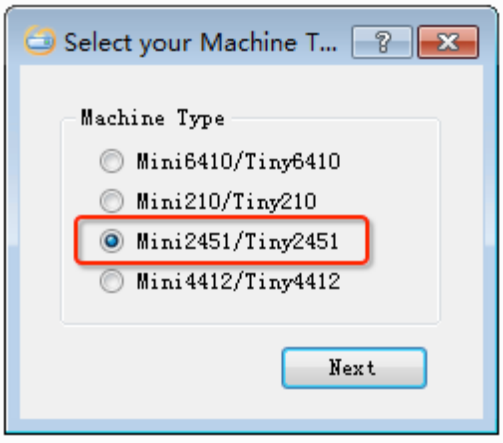
Note: users complained that some notebook's integrated SD card reader cannot work properly with card burning or reading. So far we haven't encountered this issue and we suggest that you should try a common card reader in this case.

Our SD-Flasher.exe formats a 130M space for the bootloader therefore an SD card whose memory is less than 256M cannot work and we recommend using one whose memory is at least 4G

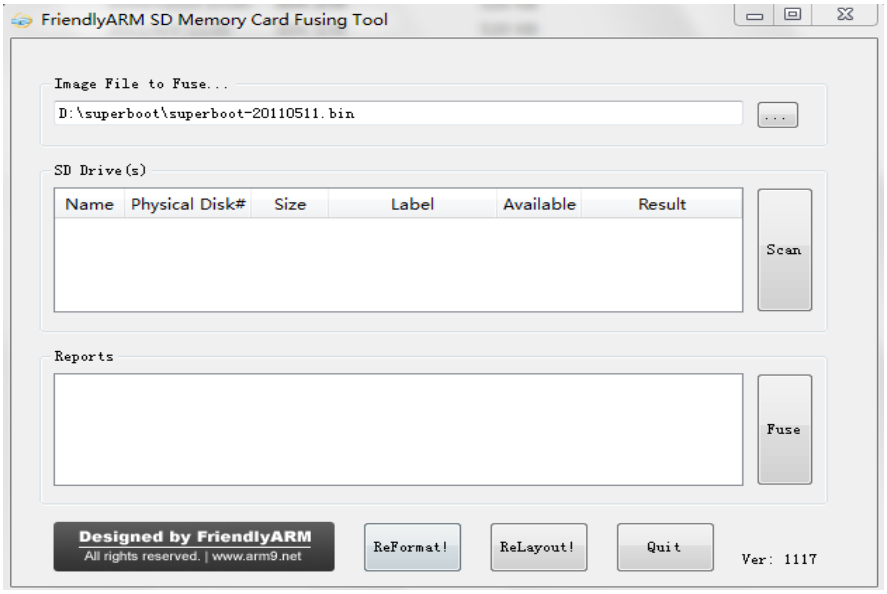
Step1: launch the SD-Flasher.exe in your shipped CD (under "\tools\"). Note: this program should be run as "administrator"



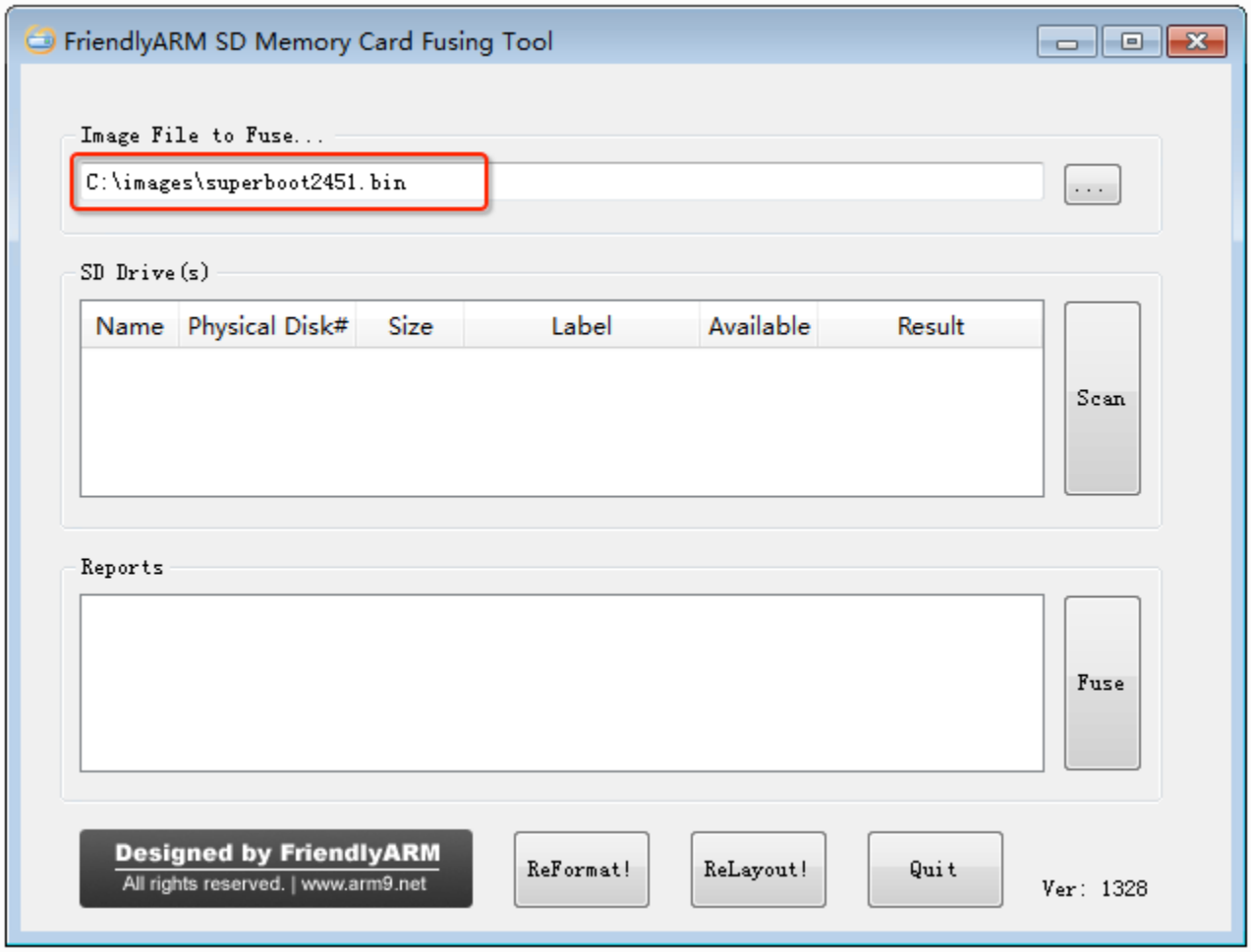
When the utility is launched a message box will pop up "Select your Machine…", please select "Mini2451/Tiny2451":



Below is the dialog you will see after it is started. Note: the "ReLayout" is enabled and we will format the SD card with this function.

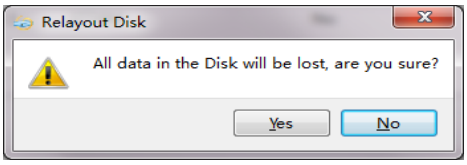


Step2: click on  to select your bootloader file

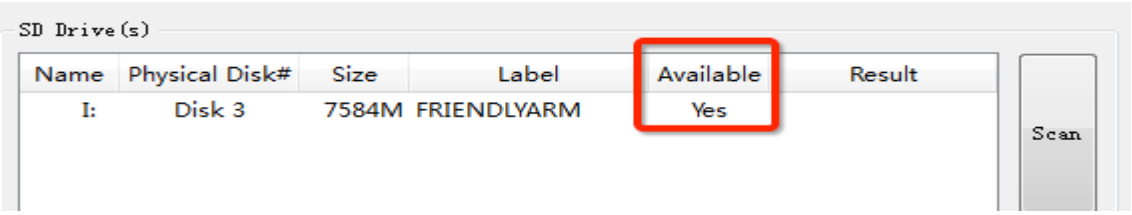


Step3: insert a FAT32 SD card into your host's SD card socket (you can also use a USB card reader to connect to a PC), backup your data in the card and click on "Scan", all recognized SD cards will be listed.

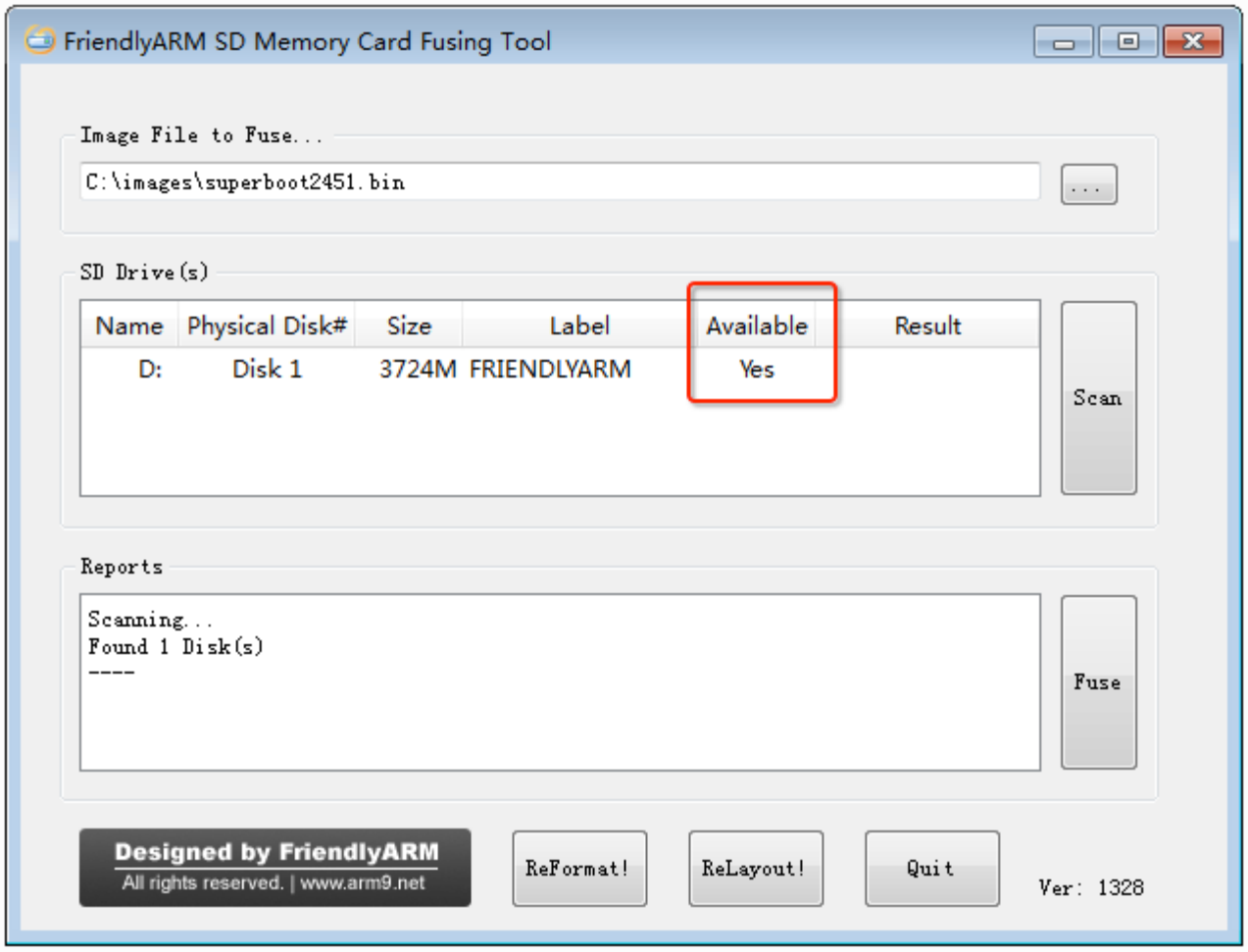
Step4: click on "ReLayout", the following dialog will pop up prompting you that the data in your card will be lost. Just click on "Yes"



After formatting is done you will be directed back to the main menu. Click on "Scan", you will see that a "FriendlyARM" section available.



Step5: click on "Fuse", Superboot will be safely burned into the SD card. You can burn this card in WindowsXP without worrying about its FAT32 data being lost or damaged.



The Superboot in your SD card is invisible. To verify it you can insert your SD card into your board's SD card socket and connect your board to a PC via a serial cable. Switch S2 to the "SDBOOT" mode, reboot your board and if messages pop up from the hyperterminal it is indicating that your SD card is functioning.

If there are no output messages from your serial port it may indicate your burning was not successful. The following cases could result in this failure:

1. You might use a notebook and the notebook's the card reader might not work. We suggest using an external usb card reader.

2. You might use a bad SD card. We suggest using one whose memory is at least 4G

or SDHC

3. You might use an SD card. Please use a standard SD card which can be directly connected to your card reader

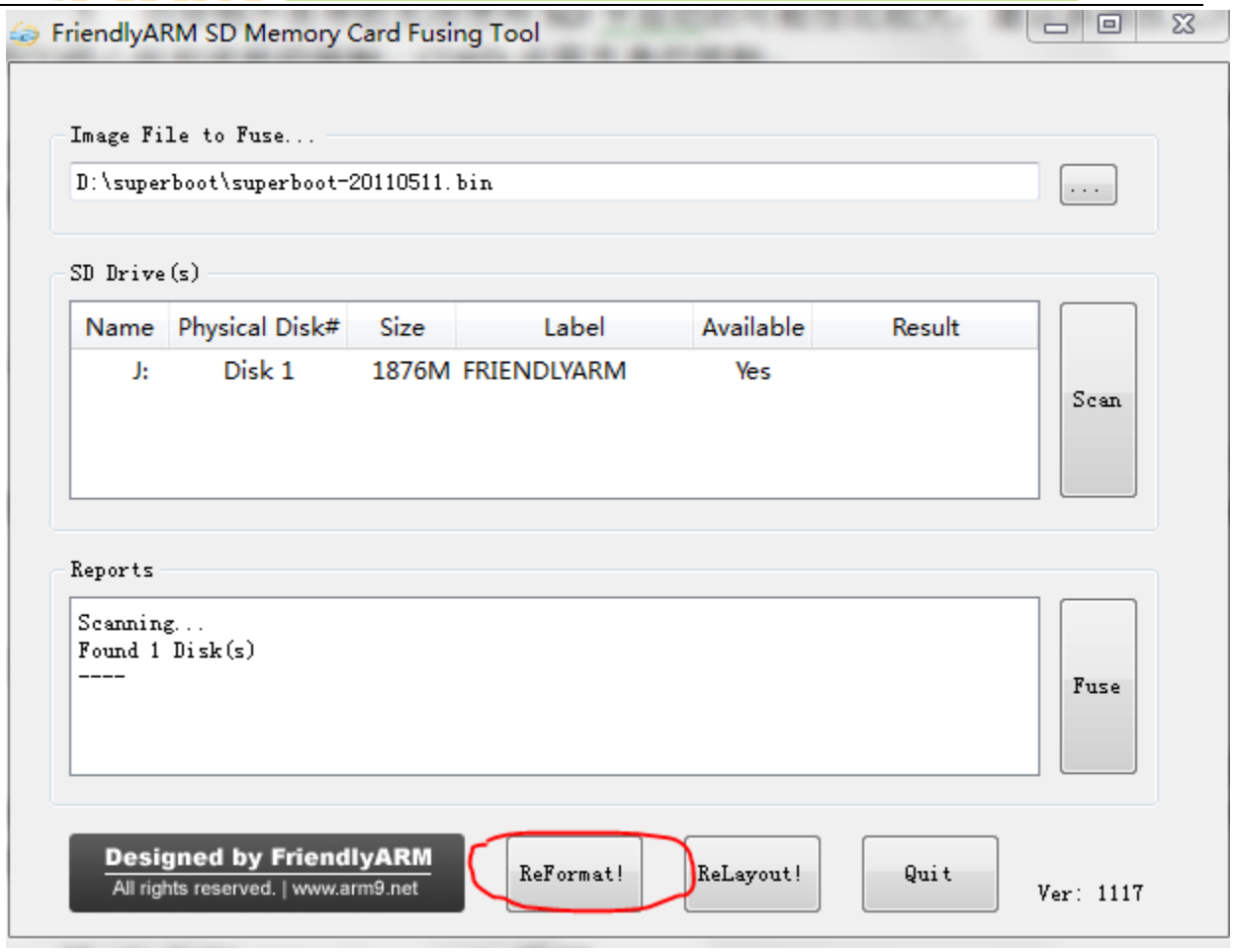
4. The SD card booting function is integrated in Samsung's chip and the ROM is preinstalled. It might not recognize some cards. In this case we suggest you try some different cards

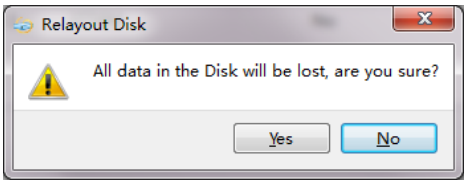
5. Poor contact might be another reason. In this case you could try a few more times: by unplugging and plugging the core board and the base board (if your board is a tiny board) and unplugging and plugging the SD card

2.2.2 Restore SD Card

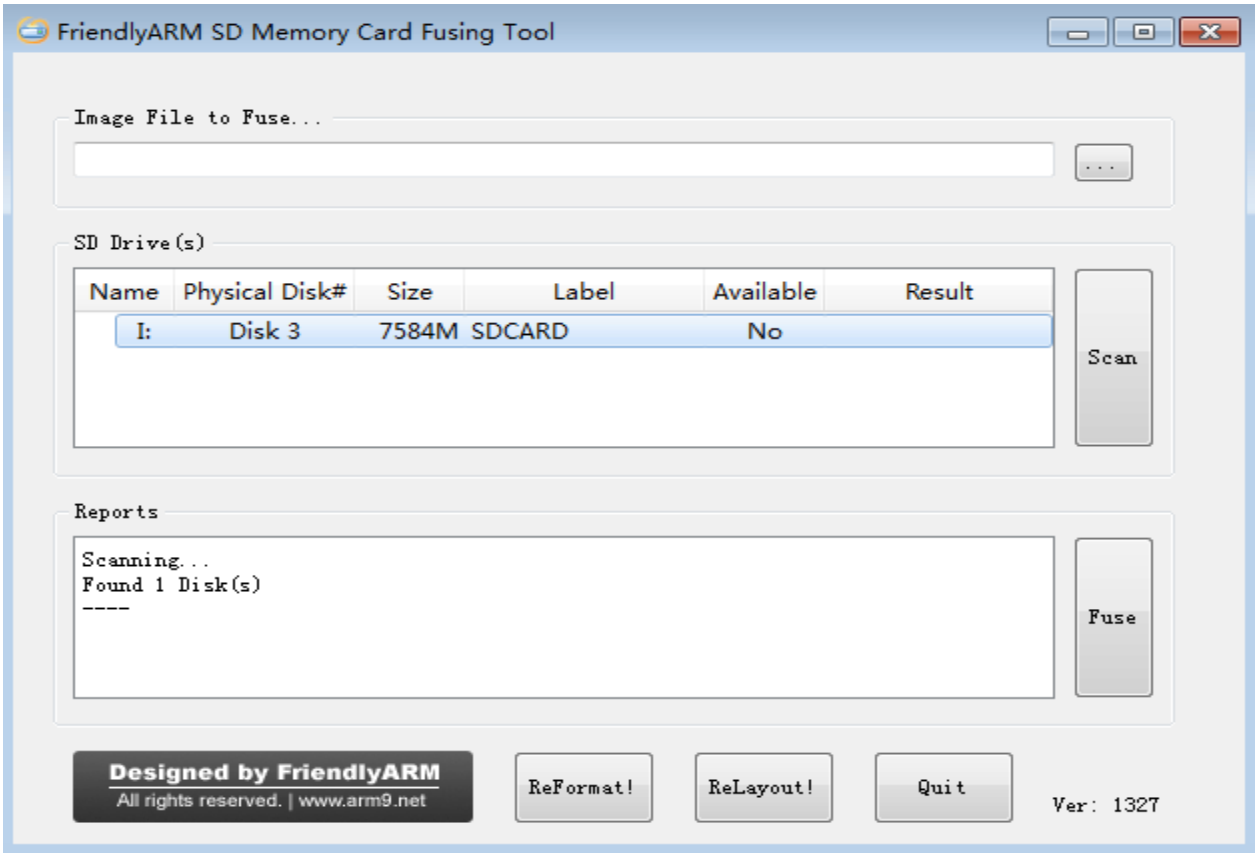
Note: we tested the following steps on Windows7

Using SD-Flasher.exe will reserve 130M memory for the Superboot. When you no longer need your SD card for system burning you might want to restore your card to what it was. You can do it this way: launch SD-Flasher.exe as an administrator; click on "scan" and "ReFormat" you will see the following dialog





Click on “Yes” and one moment later click on “Scan” again you will find your card becomes “no” available and your card is restored successfully.



2.2.3 Notes to Users

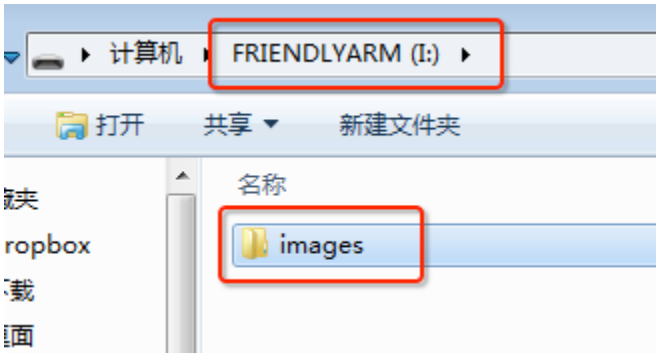
Common SD cards are used as storage cards, therefore on Vista/Windows7 SD-Flasher automatically formats an SD card to two sections: one is FAT32 (named "FriendlyARM") and the other (by default 130M) is reserved for the bootloader.

In fact, Vista/Windows7's system security policies don't permit unauthorized users to start auto burning an SD card thus common users need to format the SD card first and then burn data into it.

To burn Superboot, please run the SD-Flasher on Windows7. Running SD-Flasher on XP may not work.

2.2.4 Copy Images to SD Card

When installing systems with an SD card you need to copy the whole "images" directory to the root directory of the SD card.



2.2.5 Configure FriendlyARM.ini

When installing systems you will need the "FriendlyARM.ini" file. Its content is as follows:

FriendlyARM.ini File

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)

USBMode=Yes
Action=Install
OS=Linux
LCD-Type=W35
VerifyNandWrite=Yes
Check CRC32=No

Status Type = Beeper| LED

#Linux part
Linux-BootLoader = superboot2451.bin
Linux-Kernel = Linux/zImage
Linux-CommandLine = root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200
Linux-RootFs-InstallImage = Linux/rootfs-2451.img
```

We listed the details of each item in the table below:

Item	Comment: the default configurations are different for different systems
Action	Set actions: Install/Run/Null Install – Install to the NAND Flash Run – Run from SD card Null – No action The default setting is “Install”
OS	Operating system to be loaded: The default setting is “Linux”
LCD-Type	LCD type: W35, H43, W50 and S70 The default setting is “W35”
USB-Mode	This defines whether the board will enter the USB download mode. If it is “No” the board will be reflashed with SD card
VerifyNandWrite	= “yes”, system will verify after burning is done. This is more reliable ; = “No”, system will not verify, this takes less time. The default setting is “No”
LowFormat	Perform low level formatting on the NAND Flash The default setting is “Yes”
StatusType	Status of the burning process: “LED”,“Beeper” and “LED|Beeper” The default setting is “LED|Beeper”
Items to specify Android images, they can include directories and “/” or “\”	
Linux-BootLoader	Bootloader file The default setting is Linux-BootLoader =superboot2451.bin
Linux-Kernel	Kernel image The default setting is Linux-Kernel=Linux/zImage
Linux-CommandLine	Boot arguments - When using the yaffs2 system the suggested (default) commandline is: Linux-CommandLine = root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200 - If you want to skip the calibration step please add “skipcali = yes”
Linux-RootFs-InstallImage	File system image, now only the yaffs2 system is available. The default setting is Linux-RootFs-InstallImage = Linux/rootfs-2451.img

Notes:

1. Statements after “#” will not be executed by Superboot. Actually any character except key words can be used to comment. “#” is just widely accepted

2. To prevent our Superboot from being illegally copied we make it a rule that the first

line of the ini file cannot be edited or deleted. It is:

#This line cannot be removed. by FriendlyARM(www.arm9.net)

Note: no space or any other character after the last “)” is allowed

2.3 Install Systems with Minitools

The Minitools utility is a FriendlyARM developed USB download tool which allows users to install systems more easily and conveniently. It has the following features:

- Only need a USB cable: with the Minitools users only need a USB cable to install systems
- One key action: no need to type any command.
- Compitable with 32/64-bit OS: it can be installed on both 32-bit and 64-bit Windows systems
- Cross platform: it can be installed on both Windows and Linux systems

2.3.1 Install Minitools

2.3.1.1 Install on Windows

Double click on the “MiniToolsSetup.exe” icon in the tools directory in your shipped DVD and you will be guided to install it. Just follow the prompts and take the default options. When it asks whether you want to install the driver please go by “continue anyway”. After installation is done please unplug and plug the USB cable Windows will

prompt that it is updating drivers. After Windows' updating is done you can continue.

If your installation is successful there will be an icon on your desktop. You can double click on it to run:

The minitools' main window is shown below:

2.3.1.2 Install on Linux

We tested installing the Minitools on Fedora9/Fedora15/Ubuntu12.04 64-bit systems.

Please login and execute the installation as root. Please copy the "MiniTools-Linux-YYYYMMDD.tgz" in the "tools" directory from your DVD to your PC and untar the ball and run the "./start.sh" command to the installation.

2.3.2 Enter USB Download Mode

In order to work with the Minitools you need to enter the USB Download mode. You can enter this mode in two ways:

1. Please make sure S2 is switched to "NAND", press and hold K1, and power on your board.
2. Enter the USB Download Mode via SD booting:
   1) Flash the superboot to an SD card with SD-Flasher
   2) Create an "images" directory in the root directory of the SD card and copy the "FriendlyARM.ini" from your DVD to the SD card's "images"
   3) Open the "FriendlyARM.ini", set "USB-Mode = yes" save and close
   4) Power on the board you will see the LCD showing "USB Mode:Waiting" if everything works correctly
   5) Please connect your board to your PC via a USB cable
   6) If the connection is successful the LCD will show "USB Mode:Connected"

Now you can start installing systems with the Minitools

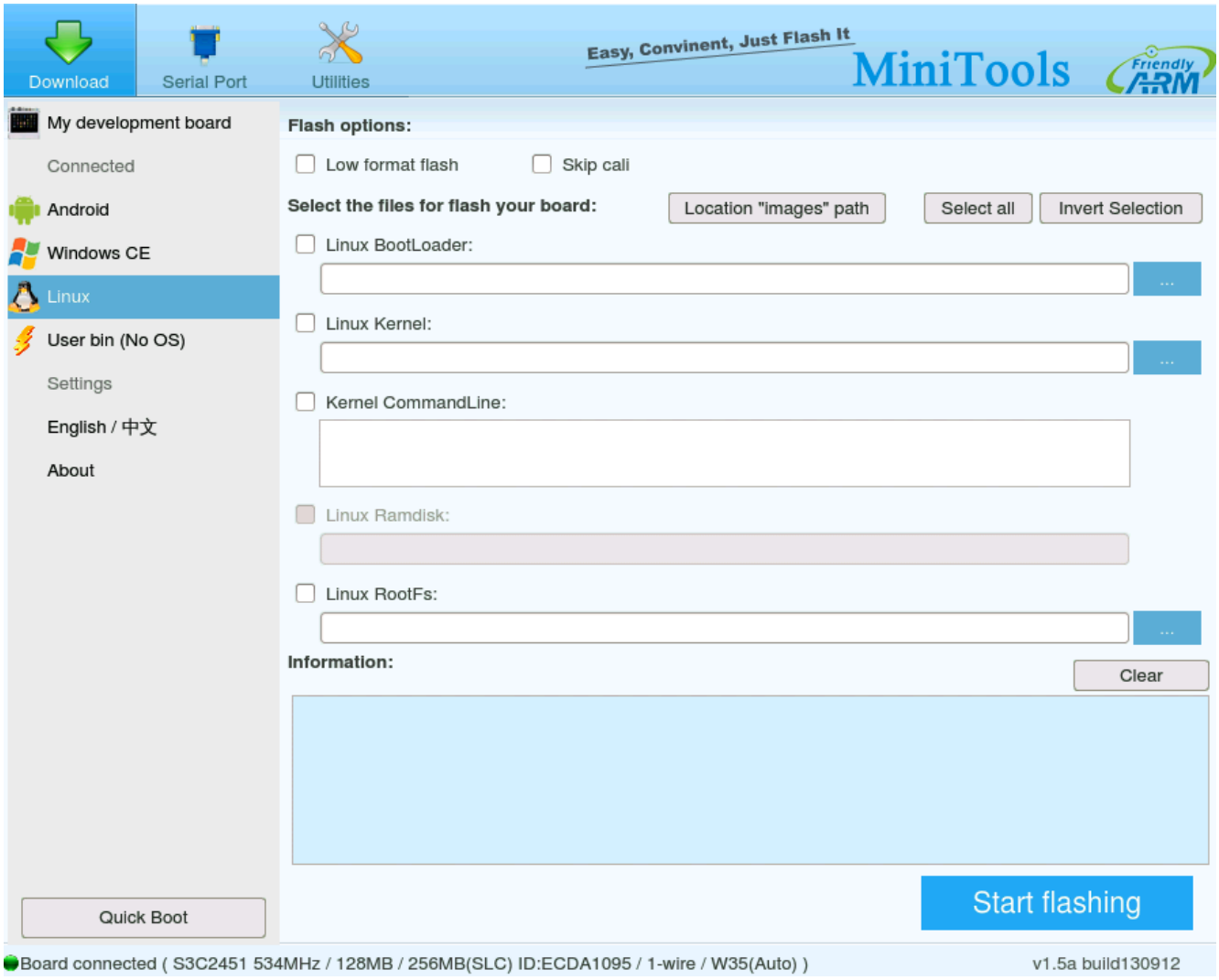
To change the installation method back to SD card installation you just need to change the "USB-Mode = yes" to "USB-Mode=no".

2.3.3 Install Systems with Minitools

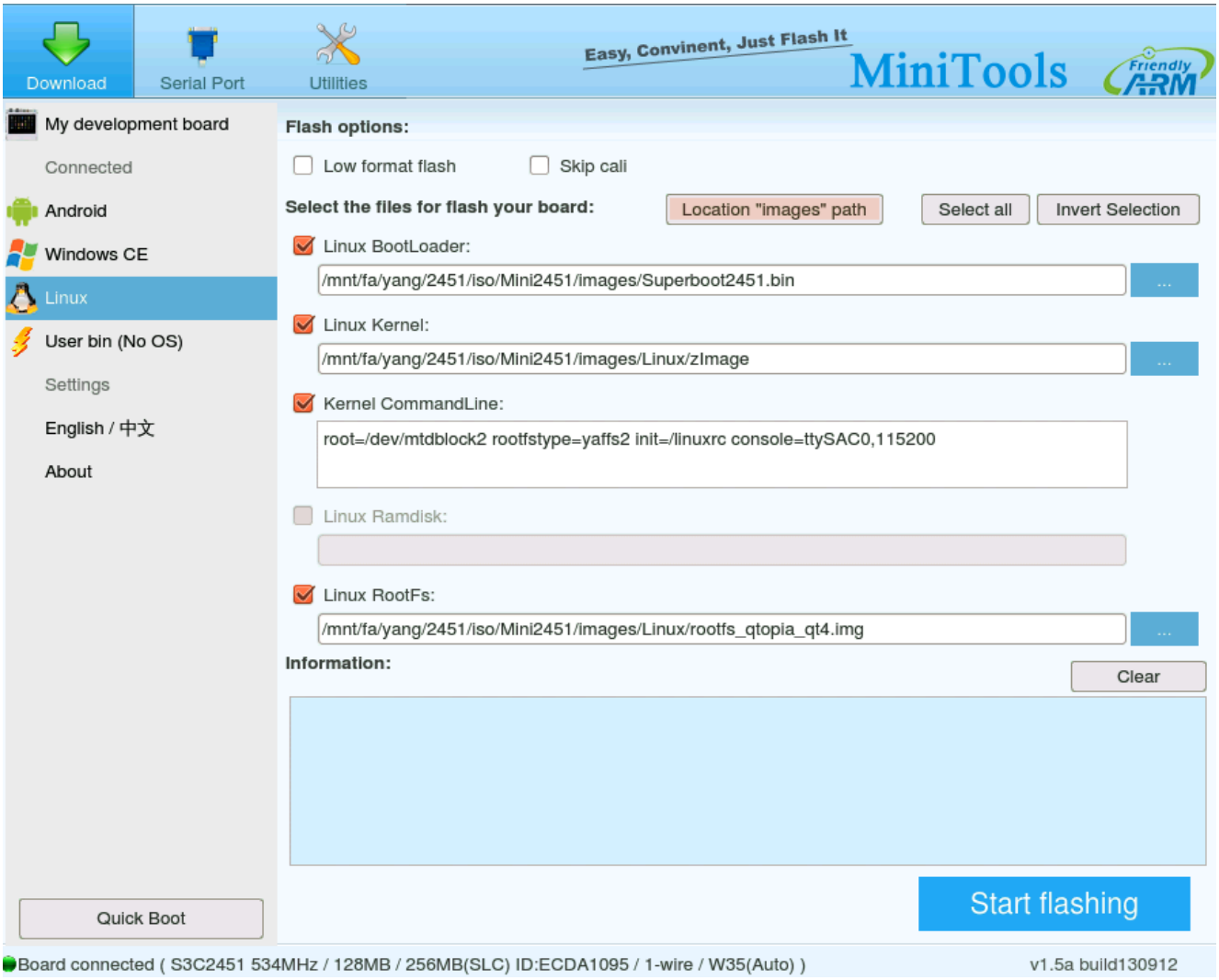
Please enter the USB download mode and connect your board to your PC which runs the Minitools via USB.

On the left bottom of the window there is an LED which is green indicating the board is connected successfully. On the left bottom there is a button which can start your board directly without switching to NAND.

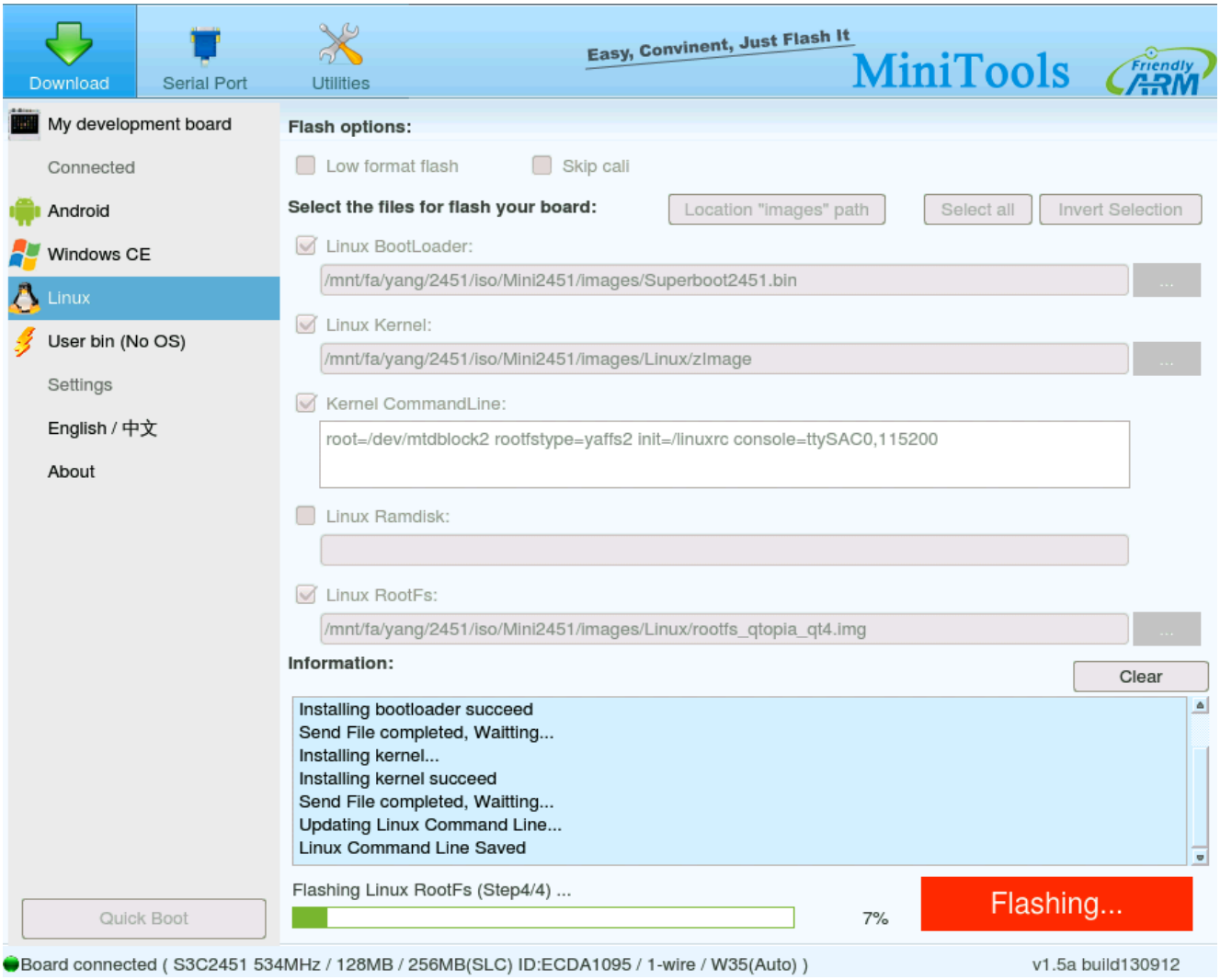
Before install systems please select the system you want to install e.g. Linux then its configuration will be presented as follows:



You can just click on the “images” button to select an “images” directory which contains complete installation files for all systems and the Minitools will show all the info listed in the FriendlyARM.ini.

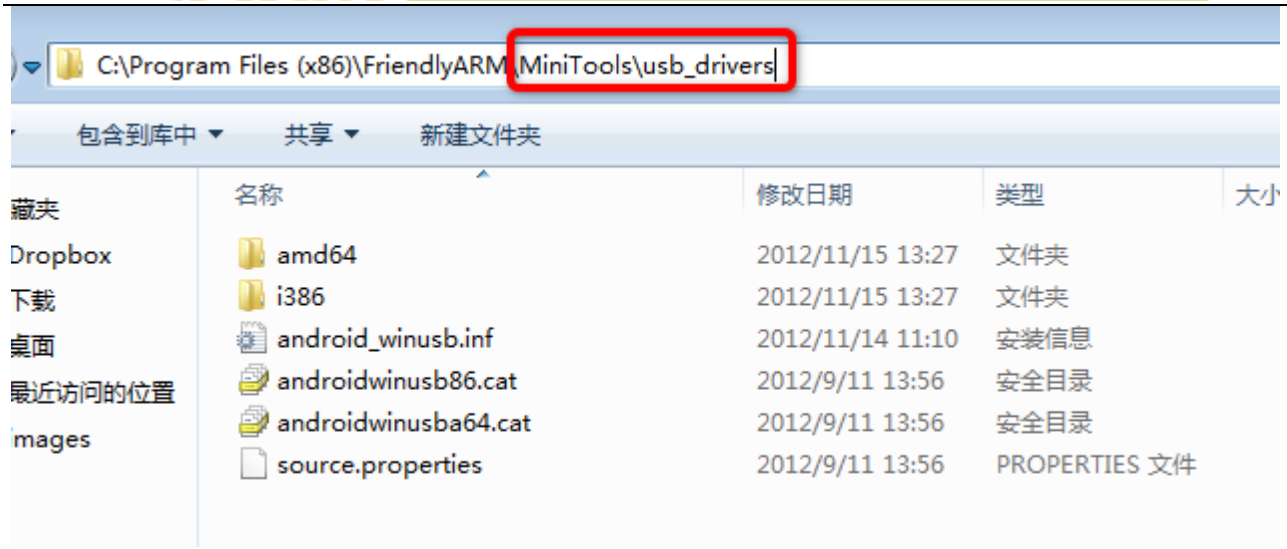


With the Minitools utility you can update either the whole system (all image files) or individual image files e.g. the kernel image file. After you are done with your installation configuration please click on “Start installation”



After installation is done you can boot your board and enter your system.

Note: sometime users complain that Minitools shows the board isn't connected to PC. It is very likely that the USB download driver is not properly installed on your PC and you can try to manually install the USB download driver which is under the Minitools directory in the shipped DVD



2.4 Install Systems with SD Card

You can flash the Mini2451 with an SD card.

2.4.1 Install Linux

Note: before read the following sections please burn Superboot to your SD card and copy corresponding installation files to your card.

Step1: insert the SD card to a PC, open the "images\FriendlyARM.ini" file and modify it as follows:

```
#This line cannot be removed. by FriendlyARM(www.arm9.net)

Action=Install
OS=Linux

LCD-Type = W35
USB-Mode = No

VerifyNandWrite=Yes

Status Type = Beeper| LED
```

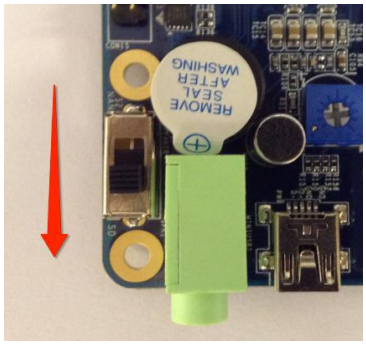
```
#Linux part
Linux-BootLoader = superboot2451.bin
Linux-Kernel = Linux/zImage
Linux-CommandLine =
CommandLine = root=/dev/mtdblock2 rootfstype=yaffs2 init=/linuxrc console=ttySAC0,115200
Linux-RootFs-InstallImage = Linux/rootfs_qtopia_qt4.img
```

(Note: currently our supported LCD types include H43, W50 and S70)

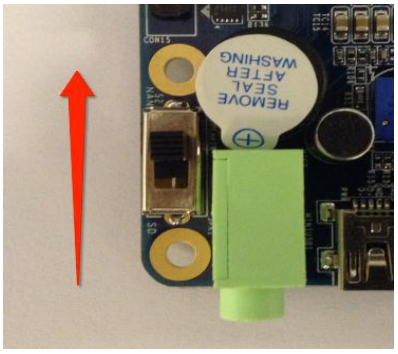
Step2: make sure your SD card has the following files (actually you can copy the whole image directory to your SD card's root directory)

File	Comment
images\superboot2451.bin	Bootloader. It can boot Linux and other OS. It can be run from an SD card.
images\Linux\zImage	Linux kernel. It can automatically detect LCD types
images\Linux\rootfs_qtopia_qt4.img	Linux file sysem image
images\FriendlyARM.ini	Configuration file

Step3: insert the SD card to the board's **SD** socket and switch S2 to the SD side. When power on the board you will hear a beep and see a progress bar on the LCD.



Step4: after system burning is done you will hear two continuous beepings and the LCD will show the burning status. Switch S2 to the Nand Flash side, reboot the system you will see Linux loading.



If you are running Linux for the first time you will see the following calibration screen:

Click on “+”, follow it till the end position and Linux will resume.

2.5 Navigate Linux

The Linux image we prepare for the Mini2451 includes Qtopia2.2.0 and Qt E- 4.8.5. Users can switch between these two GUIs freely. By default the Linux GUI is Qtopia 2.2.0.

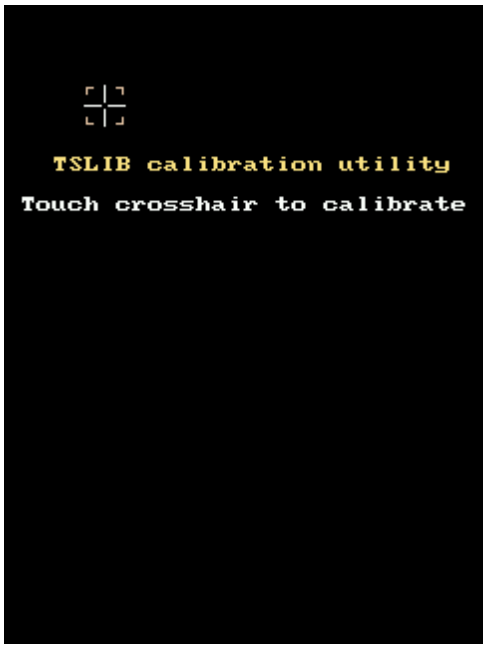
2.5.1 Calibrate Touch Screen

Note: if you didn't calibrate well you can reflash the board or use a USB mouse to calibrate after Linux is loaded.

In the following two scenarios the calibration screen will be brought up:

1. Reflash your board and restart your board for the first time

Click on "+", follow it till the end position and Android will resume. After it is completely booted you will see the following screen:



2. After enter the system go to “start->setting” and click on the “calibrate” icon you will see the above screen too.

2.5.2 Introduction to Main Pages

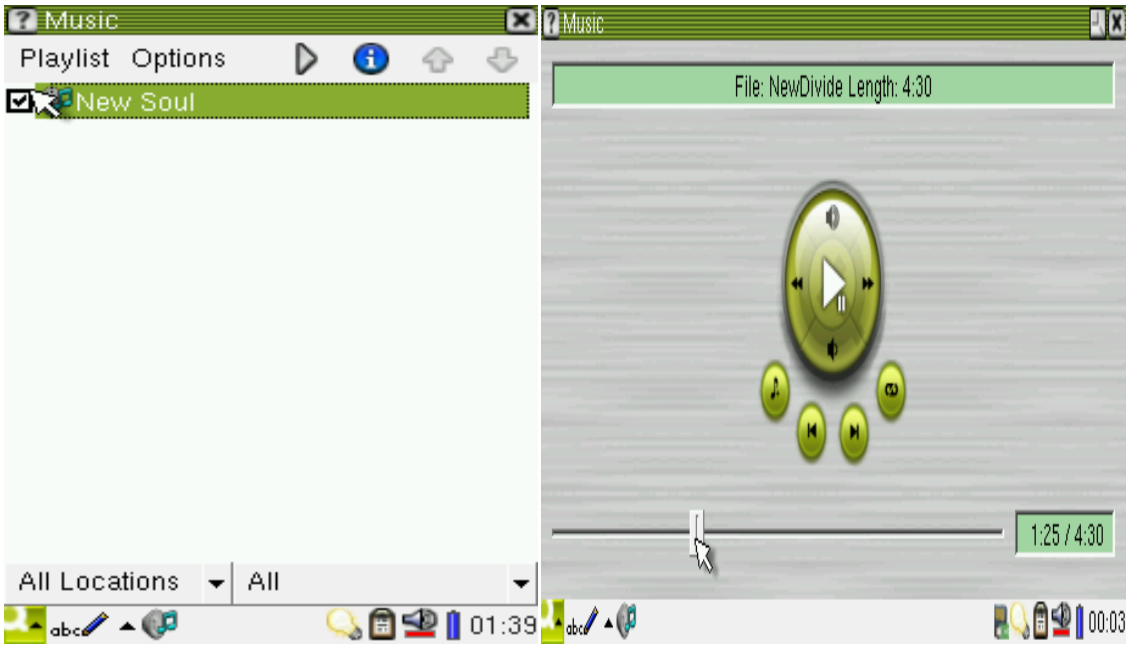
After Linux qtopia is loaded you will see the following screen.

There are five pages which represent five categories of software and utilities. Click on the “start” on the left bottom you will see five sub-menus which are the same as

these five pages. The “FriendlyARM” page contains software utilities that are all developed or migrated by us. All the other utilities and documents in other pages are open source

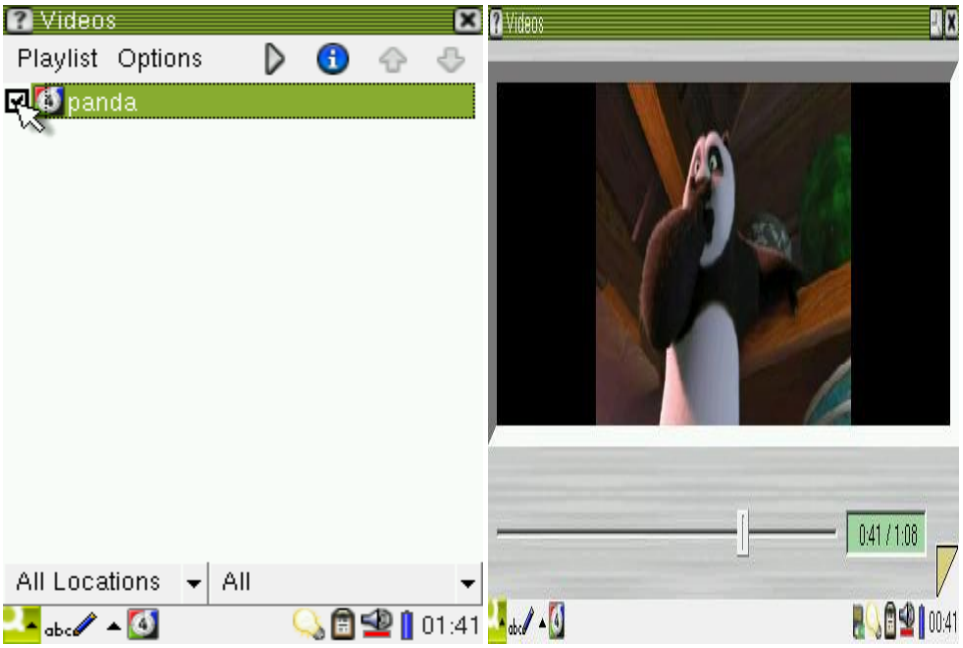
2.5.3 Play MP3

Go to the “Application” page, click on the “music” icon, select an mp3 and click on “play”.



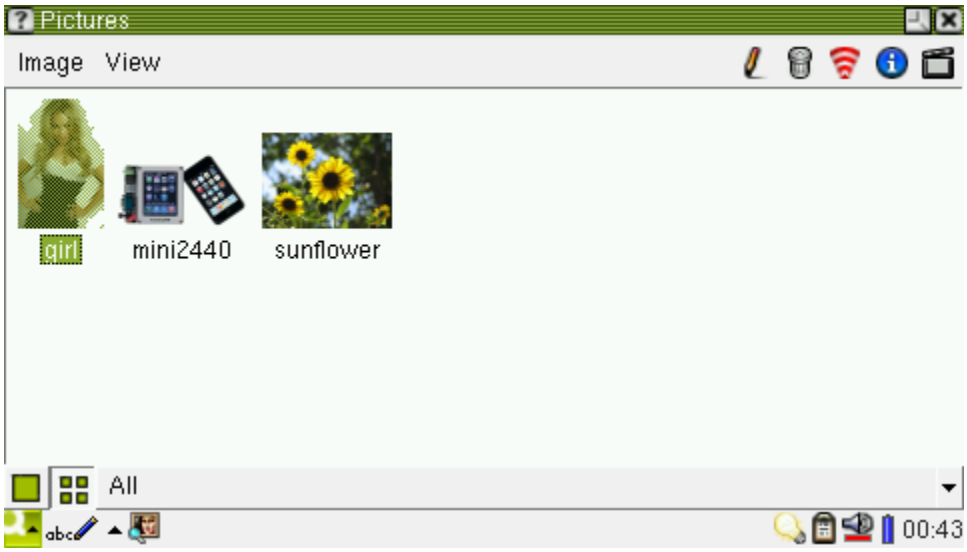
2.5.4 Play Video

Go to the “Application” page, click on the “video” icon, select a video file and click on “play”.



2.5.5 Image Viewer

Go to the Application page, click on the “pictures” icon and you will be able to browse pictures



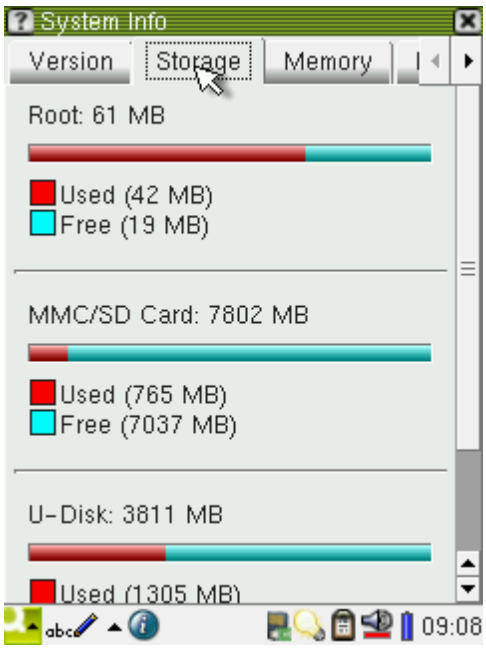
2.5.6 Auto Mount of SD Card

After your system is loaded if you plug an SD card or a USB flash drive you will see a mobile storage device icon appear on the right bottom.

All files in the MMC/SD card will be listed in the “Documents” page.

Note: this auto mount function is developed by FriendlyARM and currently it can only recognize the card’s first section and formats of VFAT/FAT32/FAT16.

Click on the “Applications” -> “Storage” you will see the card’s data

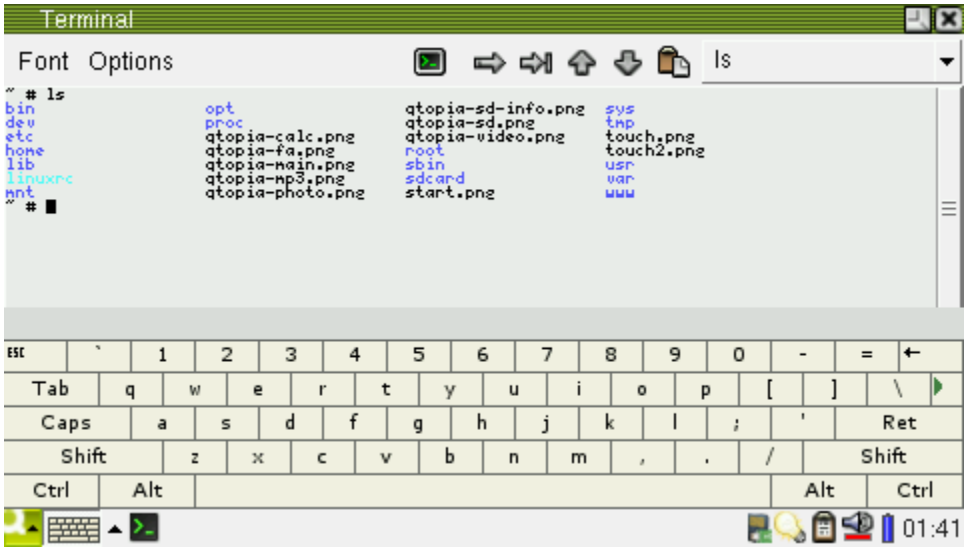


2.5.7 Calculator

Go to “Applications” and click on the calculator icon. You can select “Simple”, “Fraction”, “Scientific” and “Conversion”.

2.5.8 Terminal

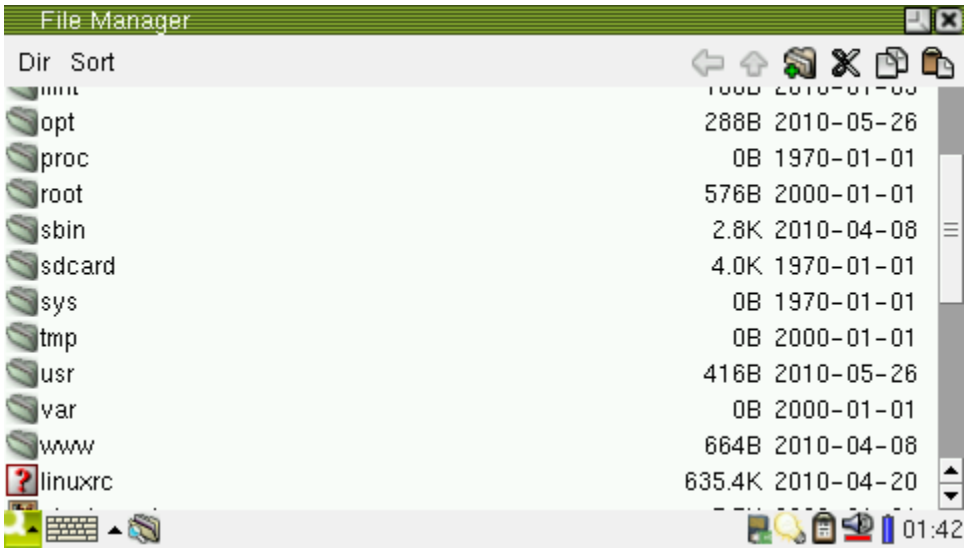
Go to “Applications”, click on the terminal icon and you will be able to type Linux commands.



2.5.9 File Manager

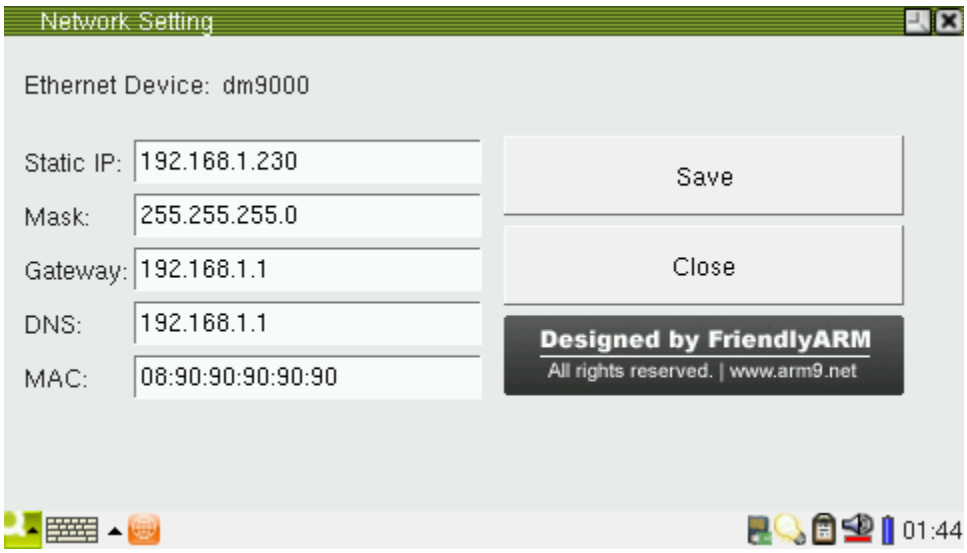
Go to “FriendlyARM”, click on the file manager icon and you will see your system’s

file structure:



2.5.10 Network Setting

Go to “FriendlyARM”, click on the network setting and you will be able to see the following screenshot



You can set your network parameters and “save” it to the “/etc/eth0-setting”.

2.5.11 Ping Test

After configure your network please go to “FriendlyARM” and click on “Ping Testing”

2.5.12 Browser

Go to “FriendlyARM”, click on “Browser”, open the soft keypad on the left bottom and you can type a website in the address bar.

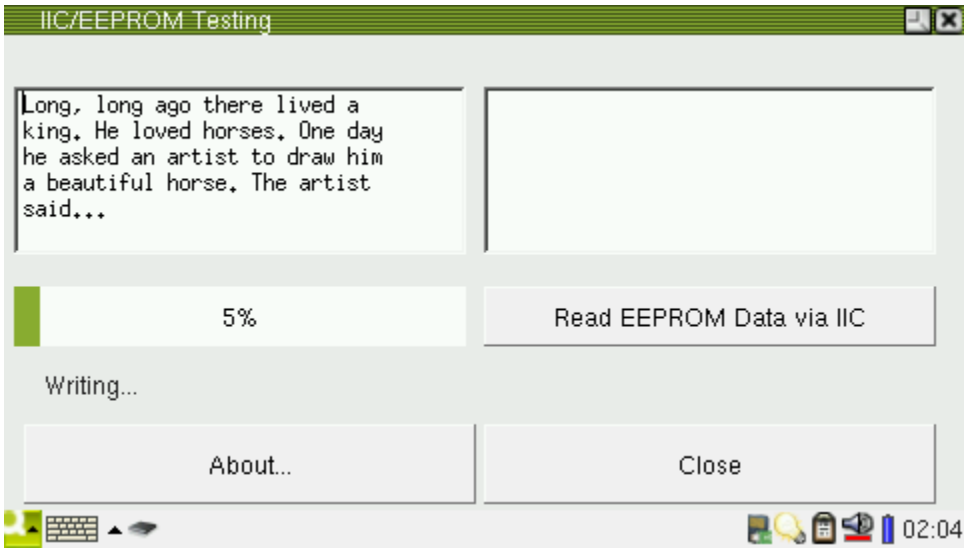
2.5.13 LED Test

Go to “FriendlyARM” and click on “LED Testing”

2.5.14 EEPROM Reading and Writing

Go to “FriendlyARM” and click on “I2C-EEPROM”. Click on “Write Data below into

EEPROM via IIC" you will see the writing process

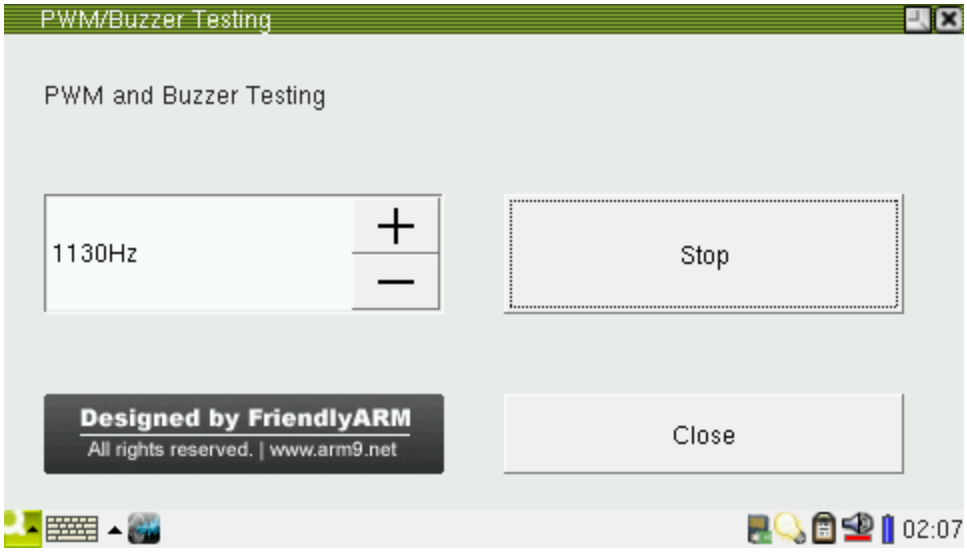


Click on "Read EEPROM Data via IIC" you will see the reading process

2.5.15 PWM Buzzer Test

Go to the "FriendlyARM" tab and click on the "PWM/Buzzer Testing" icon to open the interface. By default, the output frequency of PWM is 1000Hz. Click on the "Start"

button, the buzzer will beep. Clicking on the "+" or "-" button will change its frequency and sound as well. Clicking on the "Stop" button stops the buzzer



2.5.16 Serial Port Assistant

Note: before start this program please connect the serial port your want to test to your board.

- The on board CON1, 2, 3 and 4 are CPU UART0, 1, 2 and 3. UART0 has been converted to RS232, and extended to COM0 via DB9. On system startup it has been set to the console terminal, so it cannot be tested via this utility. The other three ports CON2, 3 and 4 must be converted to RS232 before they can communicate with a PC serial port. (FriendlyARM has a "OneCom" RS232 conversion module) When connect the ports to a PC, please make sure to use a correct serial cable (crossover serial line).

- This program also supports common USB to Serial cables. Now most laptops don't have serial ports. For the sake of users most of our agents provide those conversion cables.

Connecting a USB to Serial cable to your board, you can extend your serial ports. Its device ``` name generally is "**/dev/ttyUSB0, 1, 2 and 3**", which implies you can use a USB hub to extend your serial ports.

Connect your serial port extension board to the Mini2451's CON2/3/4 and connect to a PC via a crossover serial cable.

Go to the "FriendlyARM" tab and click on the "serial port assistant" icon to open the interface.

The title bar of the utility shows the default setting is "**ttySAC1 115200 8N1 [C]**", and it implies the default port is:

- Serial Port Deivce: /dev/ttySAC1, it corresponds to the second port UART1

- Bits Per Second: 115200

- Data Bits: 8

- Flow Control: None

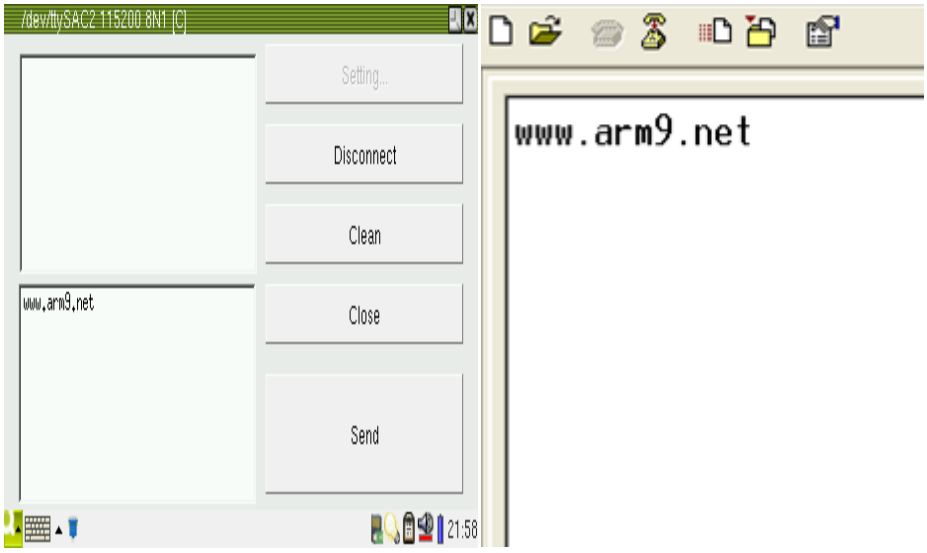
- Stop Bits: 1

- [C]: stands for the character mode;

- [H]: stands for Hex

There are two edit areas in the interface, the top one shows received data which cannot be edited; the bottom one shows sent data which can be edited via a USB keyboard or a soft keyboard.

Click on the “Connect” button to open “/dev/ttySAC1”, type some characters in the edit area, click on the “Send” button and it will send data to the connected serial port device. The screenshot below shows what a Windows super terminal receives (Note: the settings for this super terminal should be 115200 8N1)



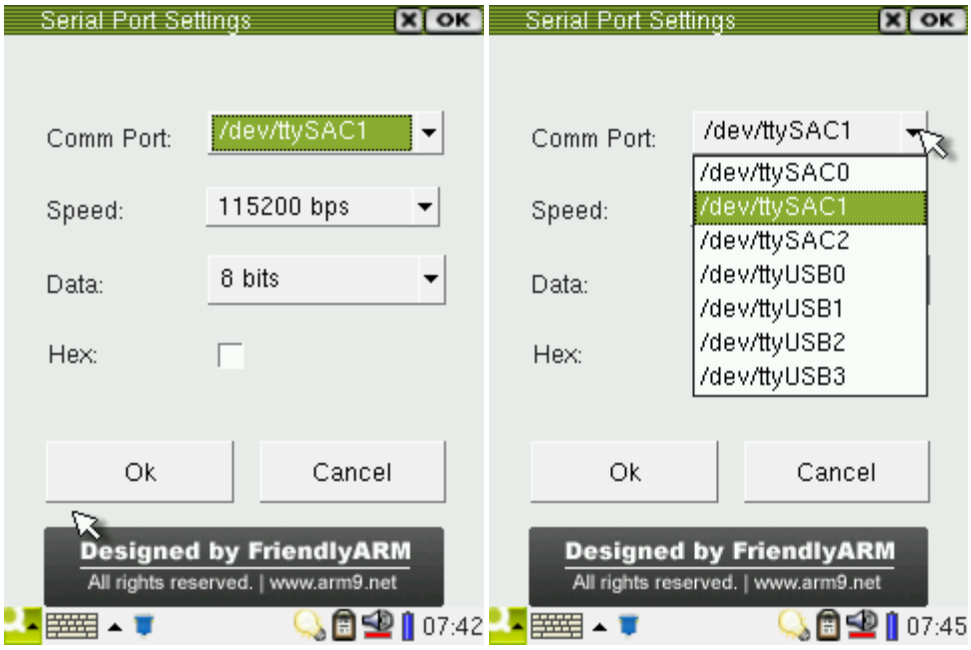
Click on “Disconnect” to disconnect the connection. Click on “Setting…” to enter the parameter setting interface which lists some basic serial port parameters:

Comm Port: you can choose "/dev/ttySAC0,1,2" or the USB to Serial "/dev/ttyUSB0,1,2,3". Note: in this utility, SAC0 corresponds to CON1, SAC1 corresponds to CON2 and etc.

Speed: bits per second

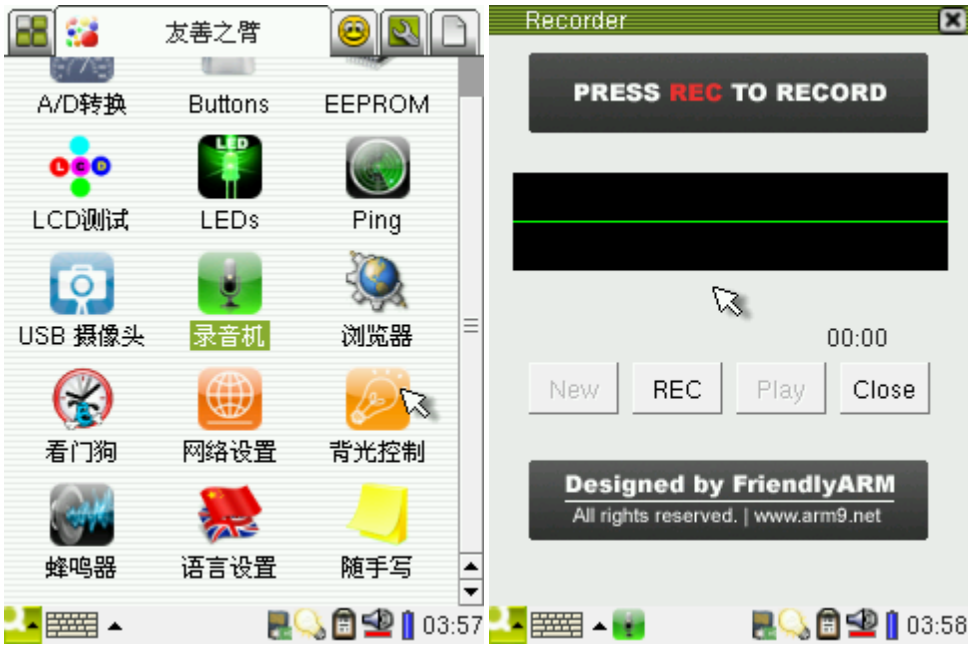
Data: data bits, 8 or 7, usually 8.

Hex: input and output data in Hex format

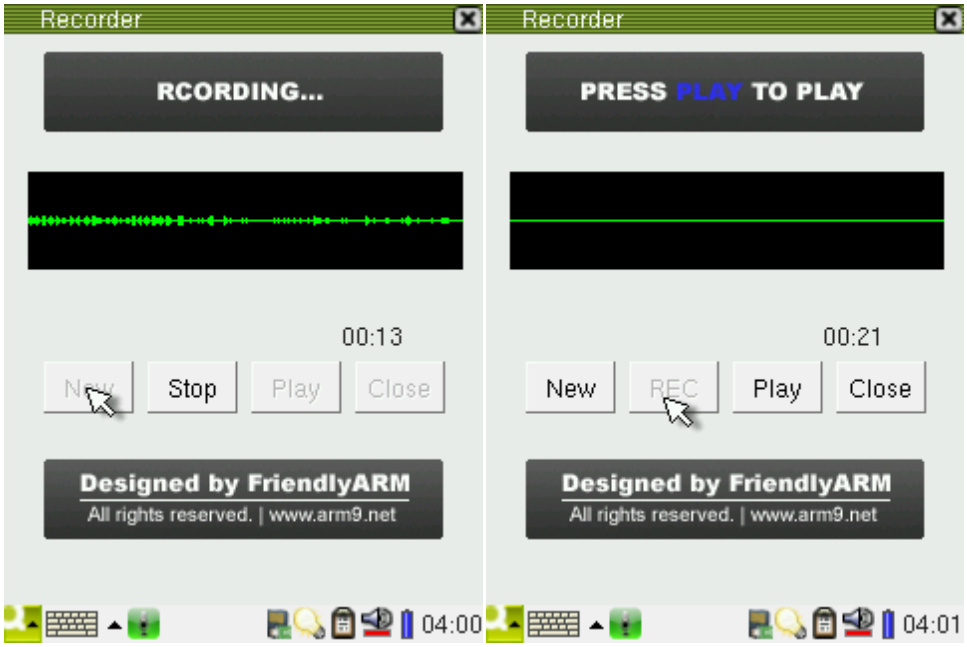


2.5.17 Audio Recording

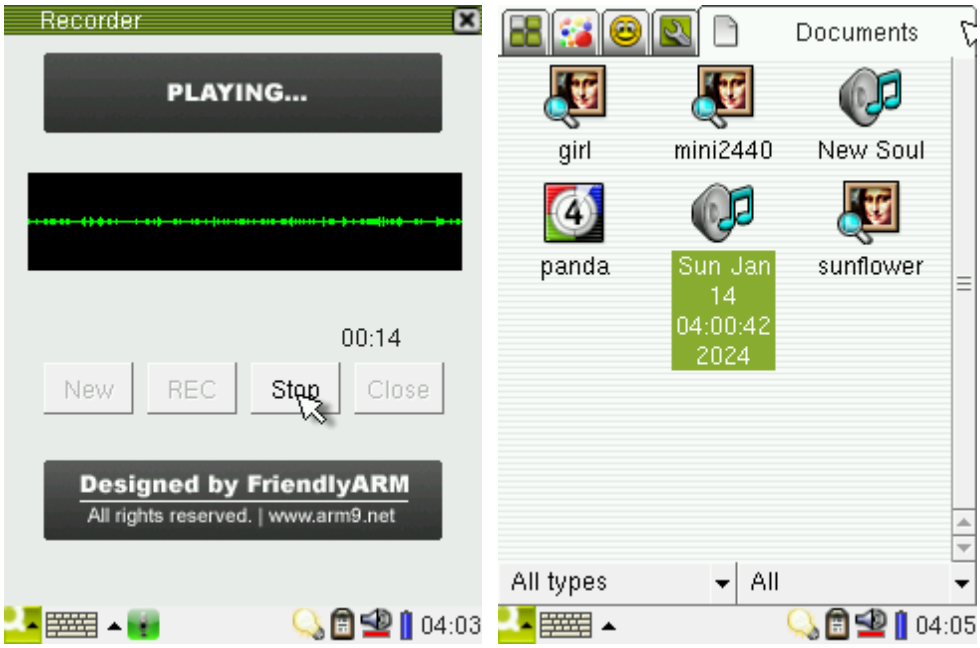
Go to the "FriendlyARM" tab and click on the "recorder" icon:



Click on the "REC" button to start recording. When you speak to the microphone on the board you will see audio waves shown on the screen. Click on the "STOP" button to stop recording.



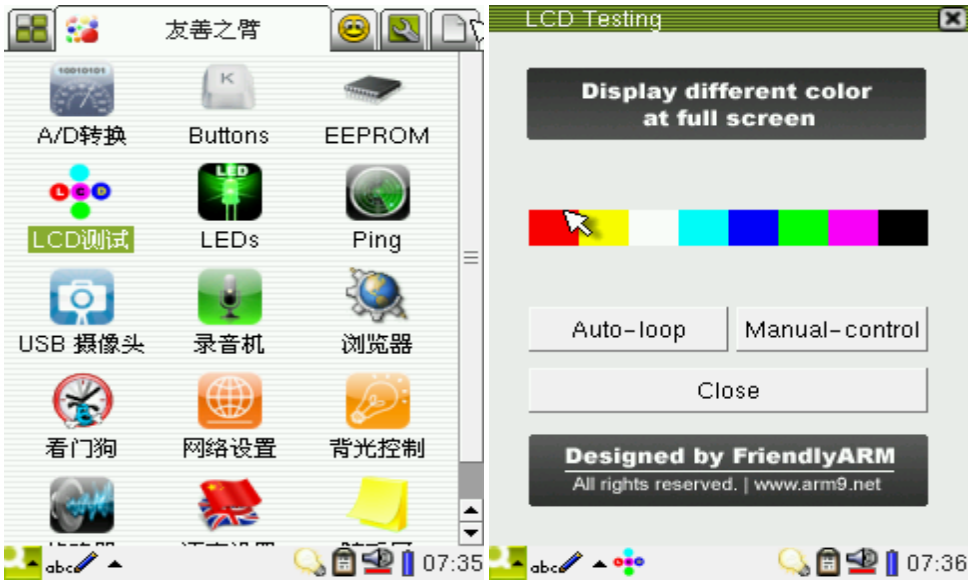
Click on the "PLAY" button to play what you recorded and you can see what you recorded has been saved as "WAV" files in the "Documents" directory.



Note: Qtopia 2.2.0 has a recorder utility by itself. But it cannot record audio. We leave it as what it is.

2.5.18 LCD Test

Go to the “FriendlyARM” tab, click on the “LCD” icon you will see the following dialog pop up:

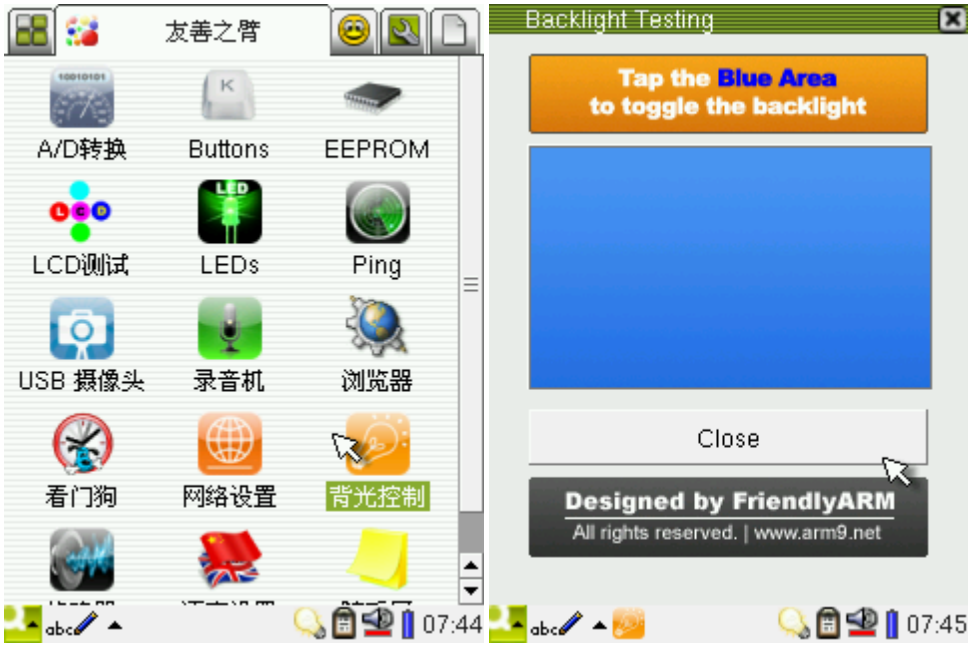


This utility has two modes: auto and manual

Auto-loop loops automatically. Executing it presents “red”, “yellow”, “white”, “sky blue”, “dark blue”, “green”, “pink” and “black”. During the loop clicking on any place on the screen will return

2.5.19 Backlight Control

In the “FriendlyARM” tab clicking on “Backlight Control” will start this utility

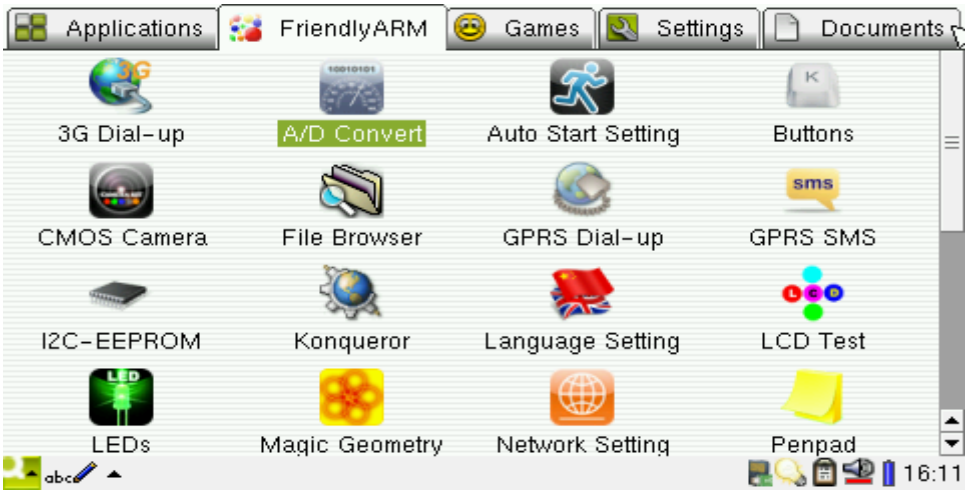


2.5.20 A/D Conversion

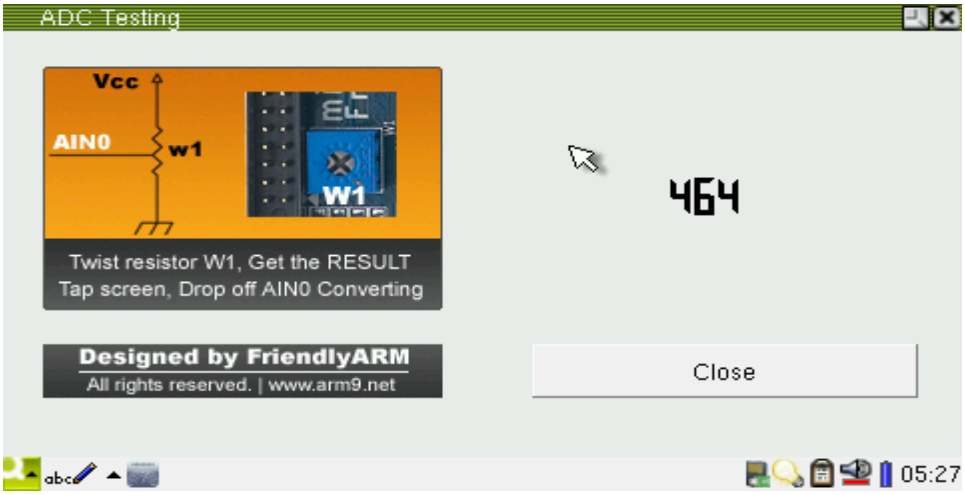
The Samsung S3C2451 chip has 10 A/D conversion channels but only one converter. In general, AIN6, AIN7, AIN8 and AIN9 are used as YM, YP, XM and XP channels via a four wire resistor. We extended AINs 0-3 which reside on CON5. For easier testing,AIN0

is directly connected to an adjustable resistor W1. How do they share a common converter? The following screenshots will show you:

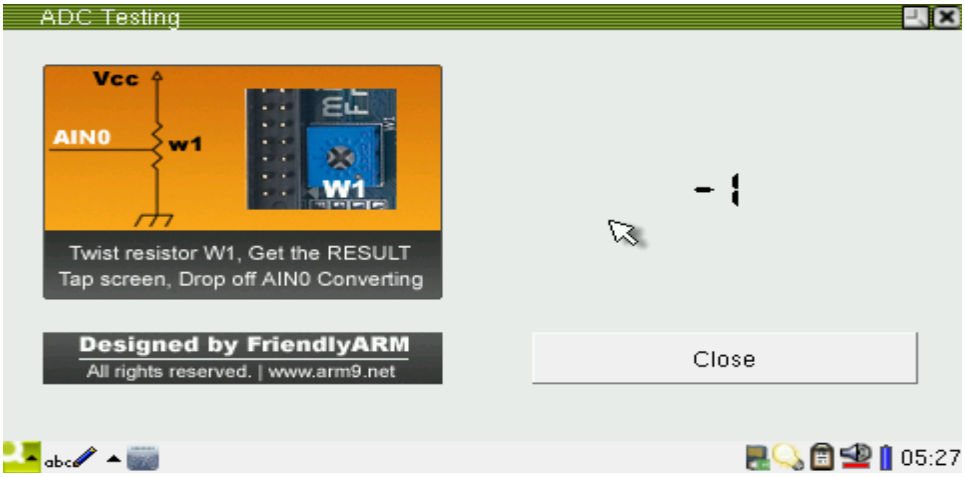
Click on the “ADC Testing” icon in the “FriendlyARM” tab:



Turning the W1 adjustable resistor, you will see the conversion changes. It has 10 digit precision, therefore the minimum value is close to 0 and the maximum value is close to 1024.



When you click on the touch screen, the A/D converter will take the touch screen as the channel, you will see the result “-1”; when you move your touch pen away from the screen, the A/D converter will take AIN0 as the channel again.



2.5.21 User Button Test

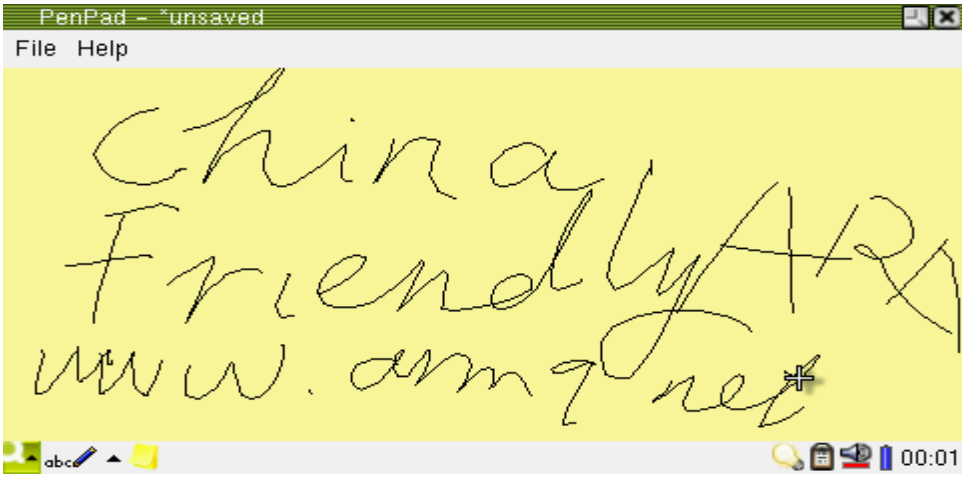
Note: the user buttons don't have dedicated functions and they are just for testing low level drivers. Click on the "Buttons" icon in the "FriendlyARM" tab. Press down any buttons on the board, the corresponding button icons will change to blue, release them, their icons will change back to grey.

2.5.22 Touch Pen Test

To test whether or not a touch pen works properly, you can draw a line on the LCD,

check if there is any offset or vibration. This can be done via the “penpad” utility. Click on the “penpad” icon in the “FriendlyARM” tab.

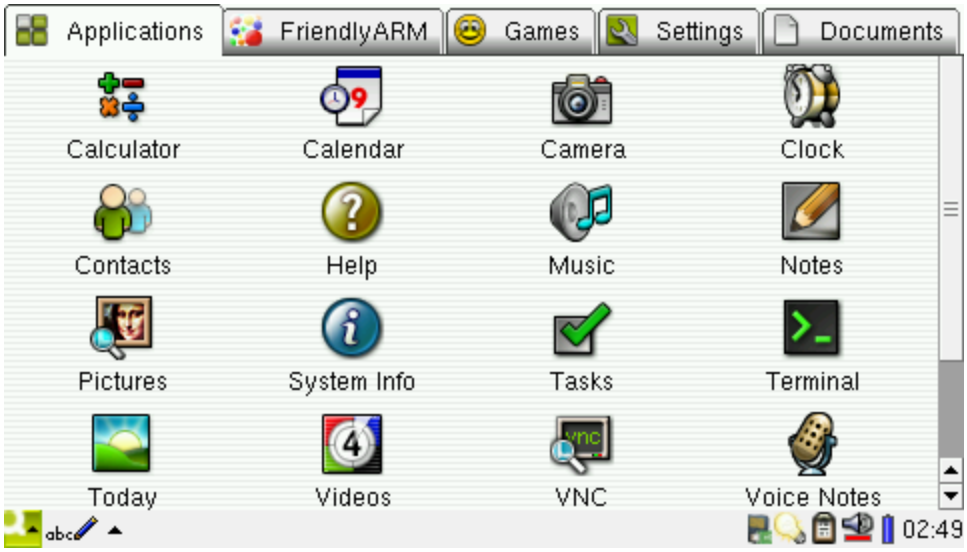
The “penpad” utility is an easy to use program developed by FriendlyARM. Start it and a yellow drawing area will show up. Draw whatever you like in the area (the pen color is black, its width is 1 pixel), go to “File” -> “Save”, you will save what you draw to a png file (in the “Documents” tab, the /Documents/image/png/ directory). The file name begins with 001. The maximum number of files that can be saved is 999. The following screenshot shows that our writing was smooth which meant our pen was accurate.



2.5.23 Language Setting

Qtopia 2.2.0 has a language setting utility which is different from the one in Qtopia 1.7.0. It only supports English. Therefore we developed a new utility located in the "FriendlyARM" tab (the icon is a waving flag).

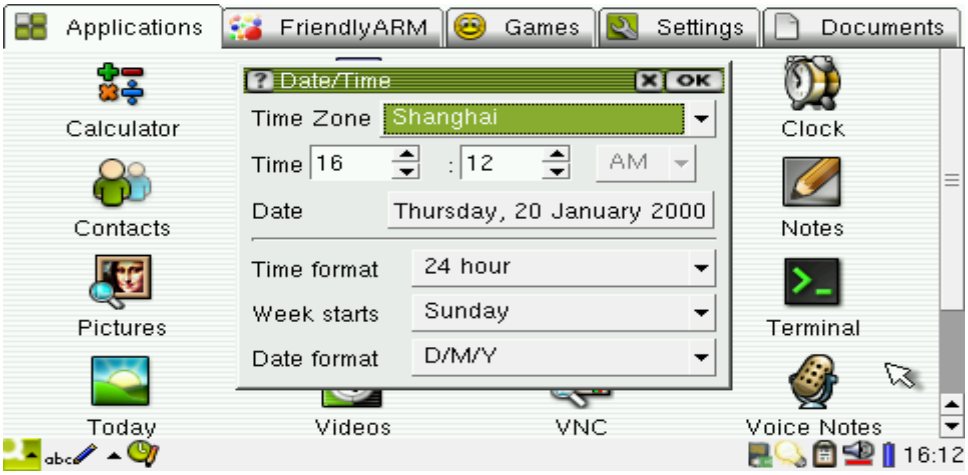
It now supports three languages: English, Chinese and Japanese. When you select "English", then click on "OK", a message will popup asking you if you want to change your language setting. Clicking on "Yes" Qtopia will reboot; clicking on "No" it will return. (Note: the Chinese and Japanese versions only have file names translated).



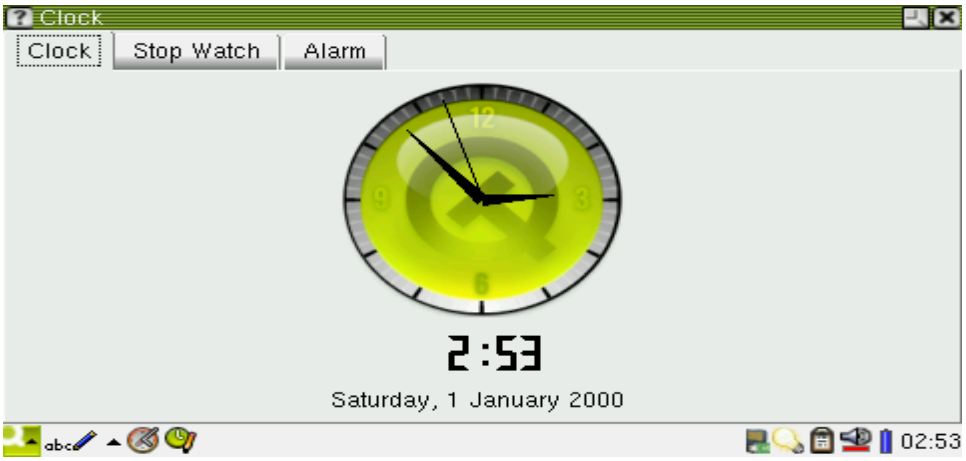
2.5.24 Time Zone, Date, Time and Alarm Setting

When you get our system, the date and time usually might not be accurate. You can adjust them by yourself. Because the CPU has its own RTC and the board has a backup battery, after you adjust the date and time, they will be saved. To adjust them, click on the time zone area at the right bottom of the screen a menu will show up. Please select “Set time..” and open the setting interface where you can set parameters such as time

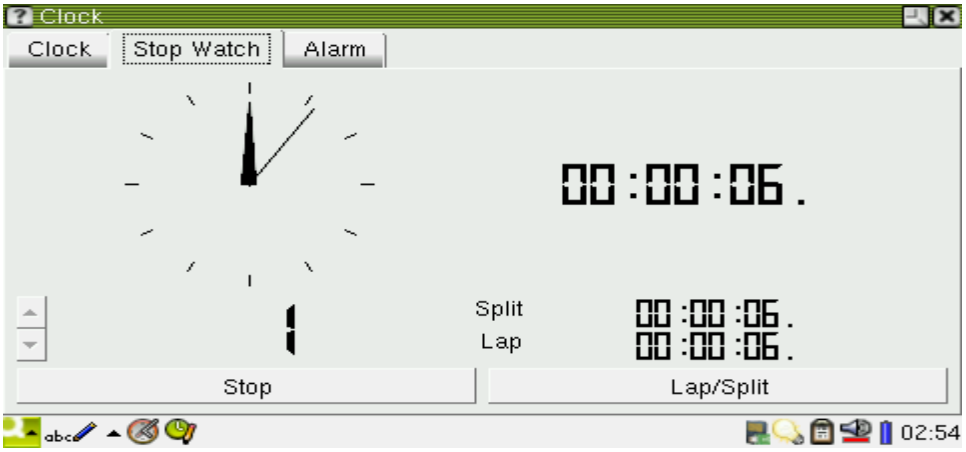
zone, date, time and so on



Select “Clock” from the menu.



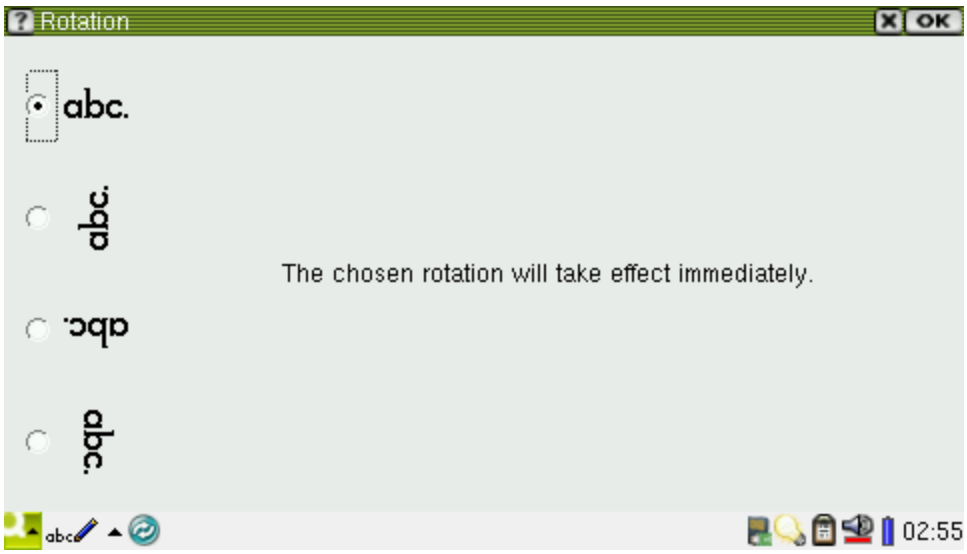
click on “Stop Watch” to open a stopwatch utility



Besides you can set the alarm clock. When it is triggered you will hear a beeping sound which lasts about one minute and the following popup window will show up. Click on "OK" to close the alarm clock.

2.5.25 Rotate Screen

Click on the "rotation" icon in the "settings" tab to enter its interface. You can rotate the screen in four directions.



Select the direction you want, click on "OK" you will see the screen rotate.

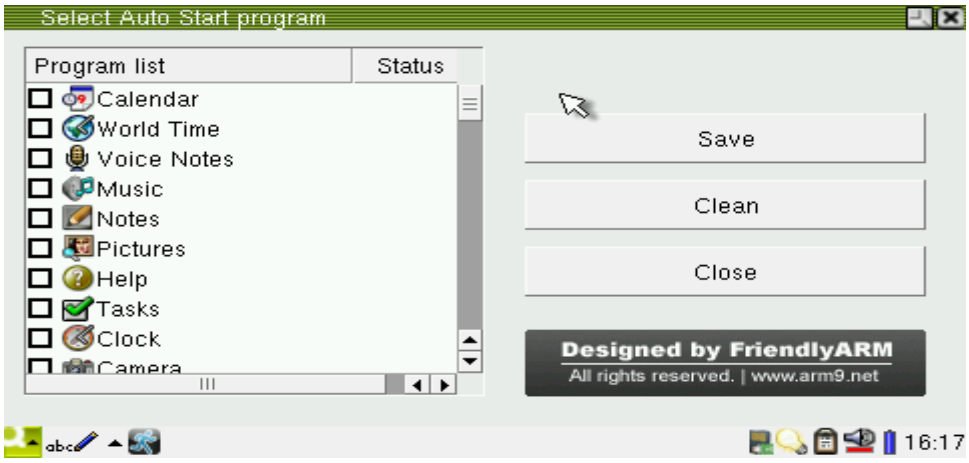
Note: sometimes you need to reboot Qtopia to see the rotation. It is a Qtopia utility and we haven't made any change to it. In addition the rotation effect is implemented via Qtopia software and has nothing to do with LCD drivers.

After rotation you will notice that all "FriendlyARM" utilities get rotated too. We implemented this feature to make our utilities displayed properly with different LCDs

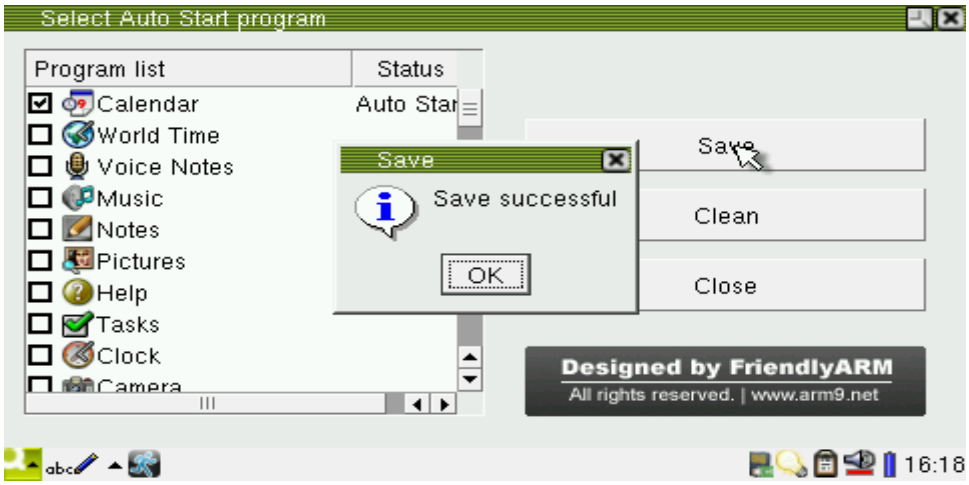
2.5.26 Set up AutoRun Program

By setting “auto run” you can make Qtopia launch its own or your programs after it boots up. It is very similar to what you see in Windows “Programs -> Startup”.

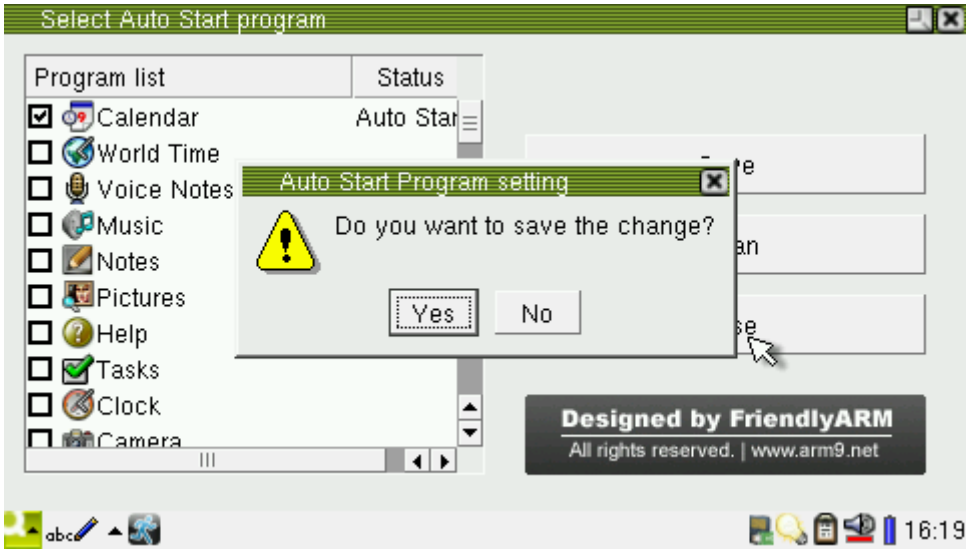
Click on the “Auto Start Setting” icon in the “FriendlyARM” tab.



Those programs listed are available programs including all Qtopia programs. The status column indicates whether a program is set to auto start. The status is unique. For instance, if the “Serial Port Assistant” is checked its status will show “Auto Start”, click on “Save”, a message box will pop up prompting that the net setting has been successfully saved. Close this utility, reboot the system you will see the “Serial Port Assistant” is auto run.



To disable auto run for a program, just click on “Clean” and “Close”, a message box will pop up, click on “Yes” the auto run for that program will be disabled.



2.5.27 System Shutdown

In the “Settings” tab, click on the “shutdown” icon you will see four options on the shutdown window.

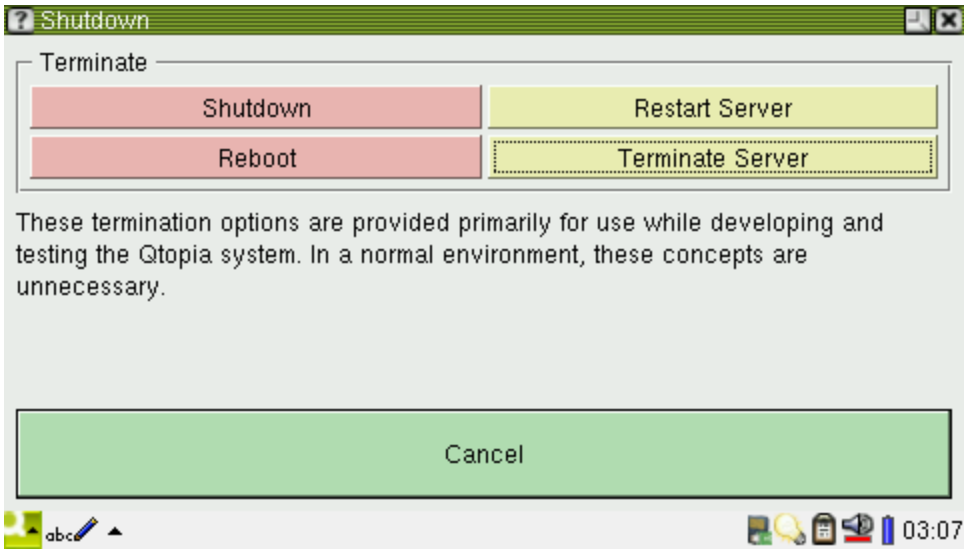
Shutdown: Press this button, Linux will end all the programs and services to shutdown the whole system. After the whole system is shutdown, the CPU will not be running and the system consumes lest power. However since our system doesn’t have a hardware power down circuit you still can see the power LED on the board is on.

Reboot: This is a “hot” reboot button. If your system boots from the Nor Flash, after you press this button, the system will shutdown, reboot and enter the supervivi main menu. If your system boots from the Nand Flash, after you press this button, the system will shutdown, reboot and enter the Qtopia interface.

Note: **Reboot** is different from the "Watchdog" function we will introduce. The "Watchdog" is "cold" reboot and doesn't end programs or services but reset the system instead.

Restart Server: it restarts the Qtopia system only. It doesn't interrupt the running Linux.

Terminate Server: it shuts down the Qtopia system. After press this button, the Qtopia interface will be disabled. What is left on the screen is the left data in RAM and it is not an active graphic interface.

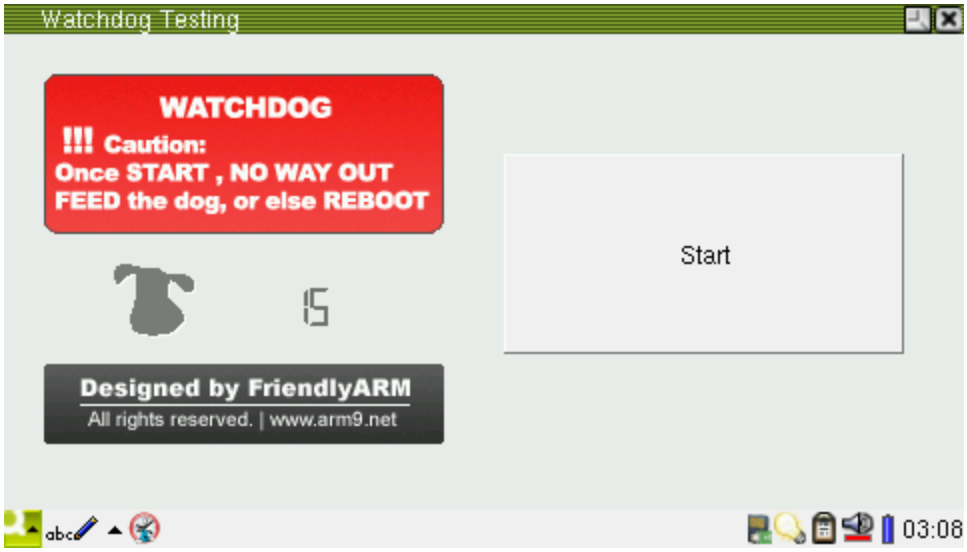


Note: the original Qtopia 2.2.0 system doesn't "shutdown" or "reboot" effectively, we changed its code to make it work.

2.5.28 Watchdog

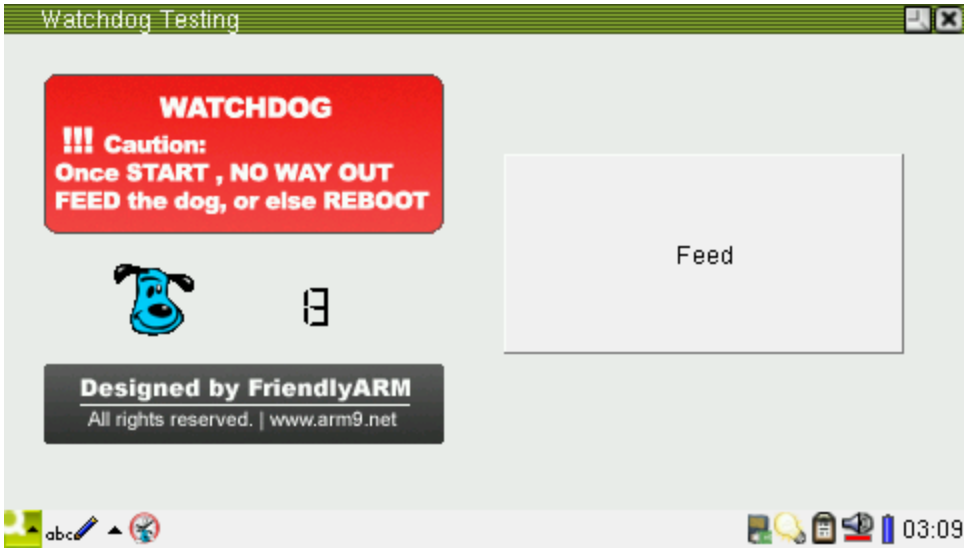
The "Watchdog" is a very basic utility in embedded systems. The S3C2451 chip already has a watchdog. The latest Linux kernel has drivers for it.

Click on the “Watchdog” icon in the “FriendlyARM” tab



Note: before take any action, please read the notes in the red area: once start, no way out, feed the dog, or else reboot!

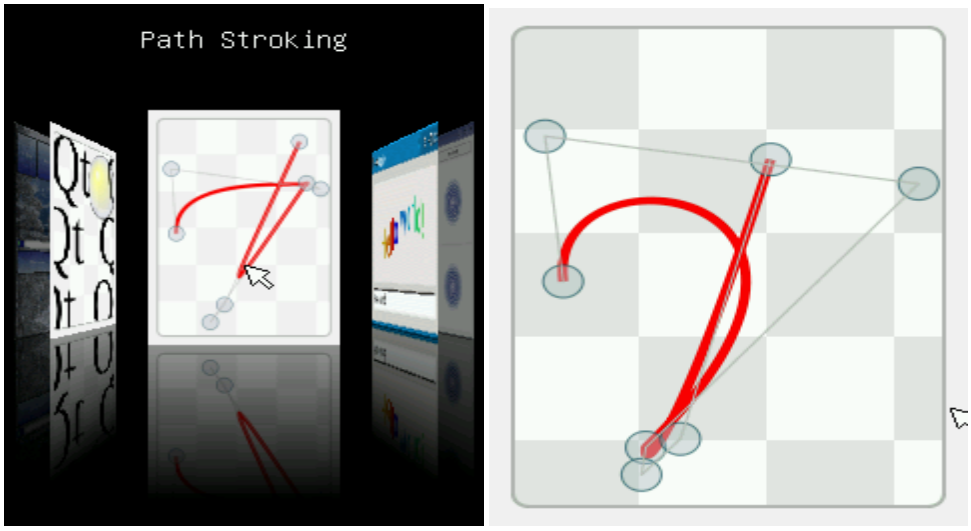
Here we set a countdown time 15 seconds. To feed the dog, click on the “Feed” button. Keep feeding, it will always have bones and the system will not reboot.



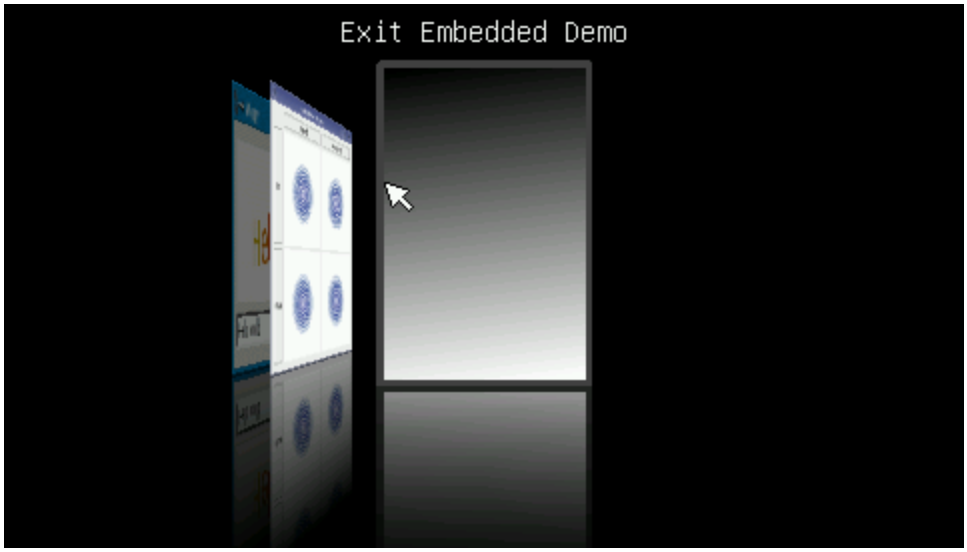
2.5.29 Start QtE-4.8.5

In order for users to switch freely and smoothly between different systems we implemented a feature that allows Qtopia-2.2.0 and QtE-4.8.5 to co-exist in the same file system. In Qtopia-2.2.0, by clicking on a common application icon users will be able to start QtE-4.8.5. After close the QtE-4.8.5 utility, users will be able to return to Qtopia-2.2.0.

QtE-4.8.5 runs as follows. It is a program manager that displays a CoverFlow effect. You can drag it left and right and run it by clicking on one of the Covers.



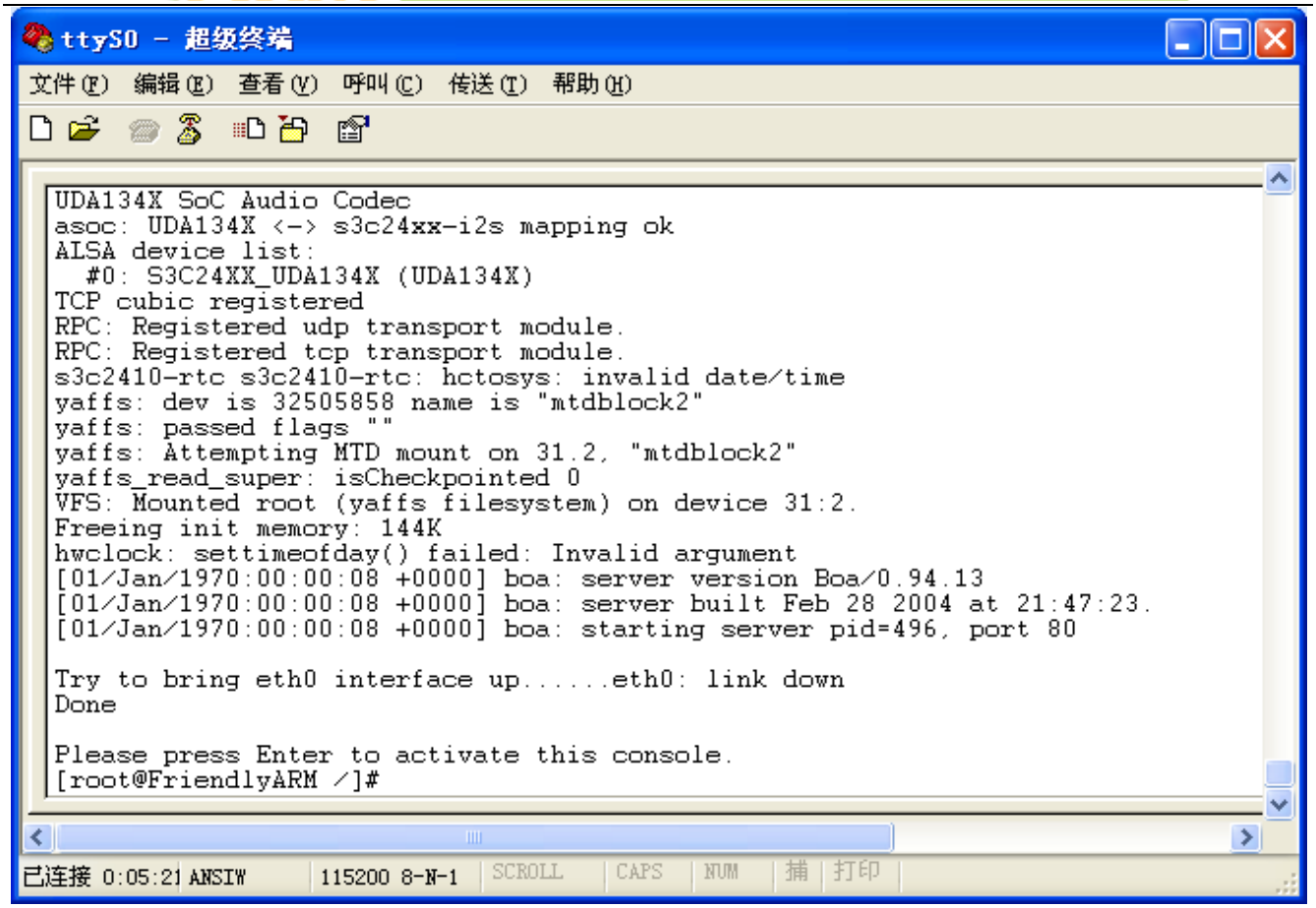
You can exit QtE-4.8.5 by clicking on “Exit Embedded Demo” and return to Qtopia-2.2.0



2.6 Navigate Linux via Commandline

Note: every Linux fan may need to get familiar with the command line utility. All Linux commands are very similar (99% of them are identical) across different versions. Before step in this section, please set up your super terminal properly.

Below is a screenshot of system login via super terminal. Just press “Enter” as prompted to continue.



2.6.1 Play MP3

The madplay utility is an mp3 player migrated by FriendlyARM. It can be run in various ways and the most straightforward one is this:

#madplay your.mp3

This command will play “your.mp3” in its default way. You can get help by running “madplay -h”. Below is a screenshot of how it works.

2.6.2 Terminate Program

To terminate a running program you can press Ctrl + C in a terminal. For instance, if you are running madplay you can press Ctrl + C to terminate it. If a program runs in the

background you need to issue the “kill” command to terminate it.

2.6.3 Mount USB Drive/Portable Hard Disk

After inserting a USB drive, the system will automatically create a “/udisk” directory and mount the drive on it, you will see the following messages:

```
Try to bring eth0 interface up......eth0: link down
Done

Please press Enter to activate this console.
[root@FriendlyARM /]# usb 1-1: new full speed USB device using s3c2410-ohci an
address 2
usb 1-1: New USB device found, idVendor=2008, idProduct=2018
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-1: Product: Flash Disk
usb 1-1: Manufacturer: Hisun
usb 1-1: SerialNumber: 020070925A001033
usb 1-1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
scsi 0:0:0:0: Direct-Access     Hisun    Flash Disk       2.10 PQ: 0 ANSI: 2
sd 0:0:0:0: [sda] 4124664 512-byte hardware sectors: (2.11 GB/1.96 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] 4124664 512-byte hardware sectors: (2.11 GB/1.96 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk

[root@FriendlyARM /]# _
```

The USB drive has a device name “**/dev/udisk**”. Entering the “/udisk” directory, you will be able to browse its contents.

Note: if your drive cannot be detected, please check whether it is FAT32/VFAT.

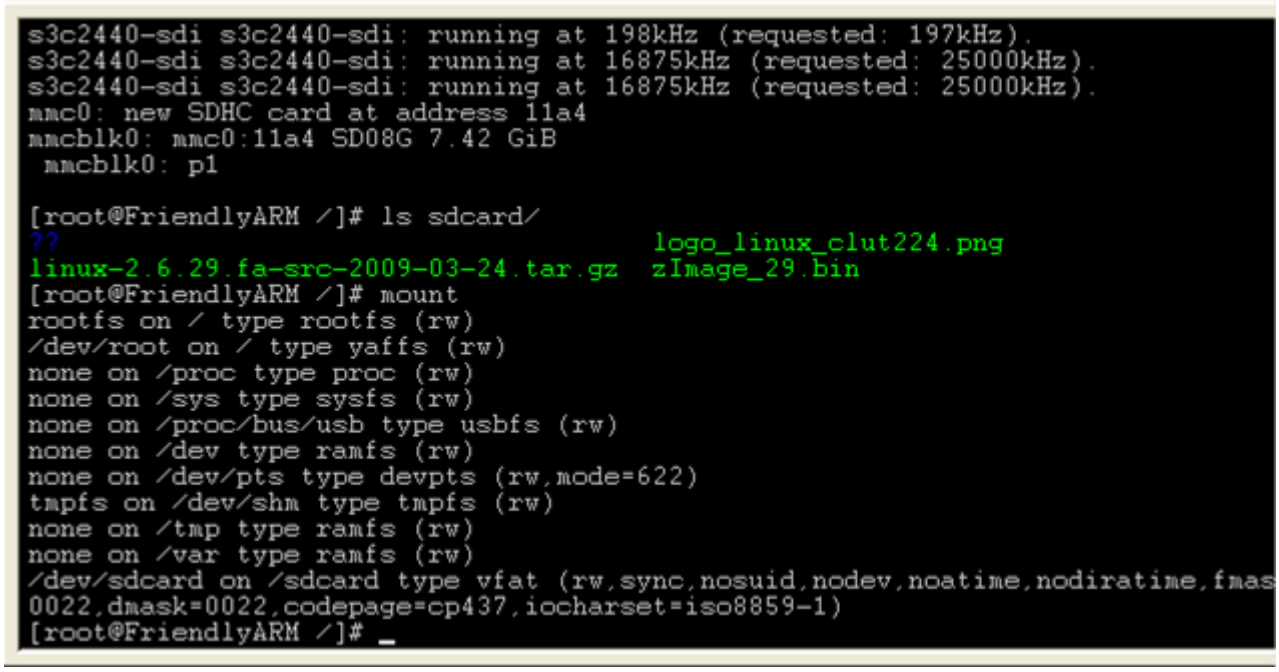
```
[root@FriendlyARM /udisk]# ls
I'm So Paid.mp3    Recycled           infinity 2008.mp3
[root@FriendlyARM /udisk]# mount
rootfs on / type rootfs (rw)
/dev/root on / type yaffs (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /proc/bus/usb type usbfs (rw)
none on /dev type ramfs (rw)
none on /dev/pts type devpts (rw,mode=622)
tmpfs on /dev/shm type tmpfs (rw)
none on /tmp type ramfs (rw)
none on /var type ramfs (rw)
/dev/udisk on /udisk type vfat (rw,sync,nosuid,nodev,noatime,nodiratime,fmask=
22,dmask=0022,codepage=cp437,iocharset=iso8859-1)
[root@FriendlyARM /udisk]# _
```

2.6.4 Mount SD Card

Similar to USB drive mounting, an SD card will be automatically detected and mounted. After inserting an SD card, you will see the following messages:

```
[01/Jan/1970:00:00:08 +0000] boa: starting server pid=496, port 80

Try to bring eth0 interface up.....eth0: link down
Done

Please press Enter to activate this console.
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]# s3c2440-sdi s3c2440-sdi: running at 0kHz (requested: 0kH
.
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 198kHz (requested: 197kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
s3c2440-sdi s3c2440-sdi: running at 16875kHz (requested: 25000kHz).
mmc0: new SDHC card at address 11a4
mmcblk0: mmc0:11a4 SD08G 7.42 GiB
 mmcblk0: p1

[root@FriendlyARM /]# _
```

The system will create a “/sdcard” directory and mount the SD card on it.



2.6.5 LED Test

(1) LED Server

After the system starts up it will automatically start a LED service (/etc/rc.d/init.d/leds). It actually runs a led-player script. After the led-player script is run a pipe file led-control will be created in the /tmp directory.

Users can change an LED's flashing by setting its parameters

#echo 0 0.2 > /tmp/led-control

After this command is executed each of the 4 LEDs will be flashing one by one with 0.2 second in between.

#echo 1 0.2 >/tmp/led-control

After this command is executed 4 LEDs will be running one by one with 0.2 second in between.

#/etc/rc.d/init.d/leds stop

After this command is executed all 4 LEDs will be turned off.

#/etc/rc.d/init.d/leds start

After this command is executed all 4 LEDs will be turned on.

(2) Manipulating a Single LED

The /bin/leds utility can be used to manipulate a single led. To launch this utility, users need to stop the led-player service first:

#/etc/rc.d/init.d/leds stop

This command will stop the led-player service. To get more information for the usage of "led" you can type the following command:

[root@fa /]# **led**

Usage: leds led_no 0|1

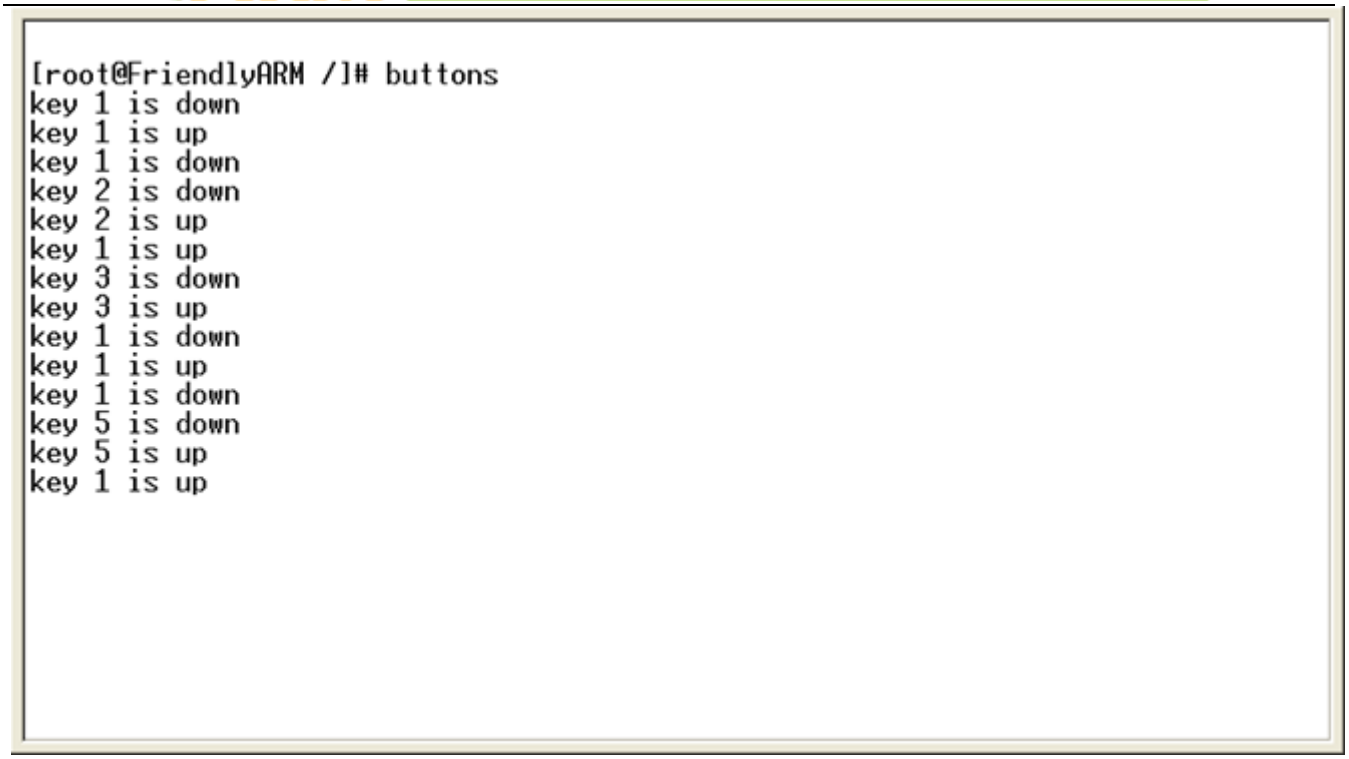
led_no: the LED you want to manipulate (0/1/2/3). "0" and "1" represents "turn off" and "turn on" respectively

#led 2 1

This will turn on LED3

2.6.6 User Button Test

Type the "**buttons**" command, press a user button and you will see the following scenario



2.6.7 Serial Port Test

Note: the armcomtest utility is a straightforward and easy to use program developed by FriendlyARM for Linux. It doesn't rely on system calls or hardware. After Linux is loaded Serial Ports1, 2, 3 and 4 correspond to **/dev/ttySAC0, 1, 2 and 3.**

To test Serial Port 2 you need a PC with a serial port. Please connect CON2 to the PC via our extension board. Type the following command:

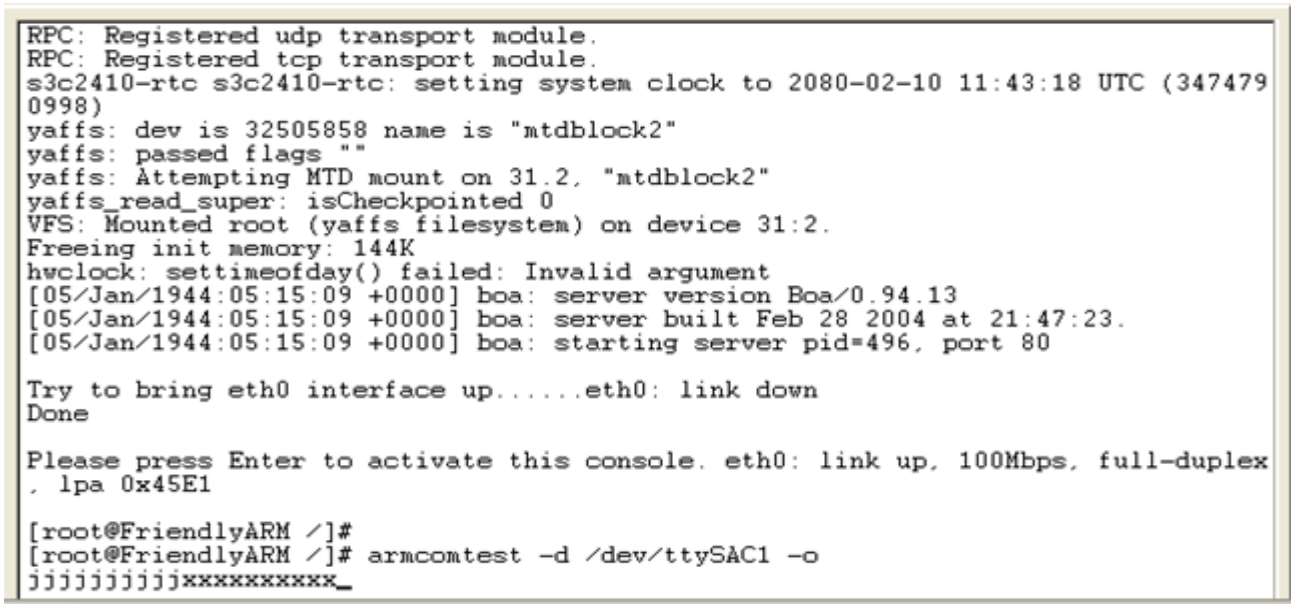
#armcomtest –d /dev/ttySAC1 -o

Now if you type characters (in Serial Port Assistant) on your board they will be output to your PC's super terminal simultaneously and vice versa

To test Serial Port 3 you need to connect CON3 via our extension board and type the command below:

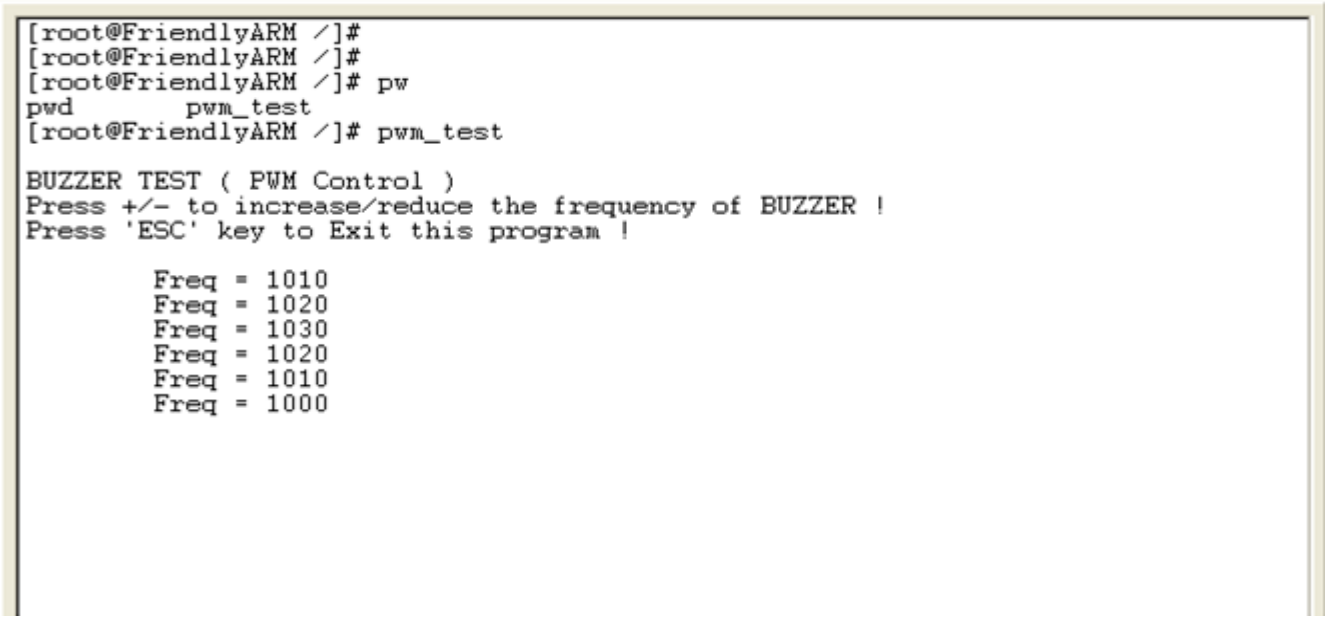
#armcomtest –d /dev/ttySAC2 -o

Here is a screenshot



2.6.8 PWM Buzzer Test

Type “pwm_test” in a terminal and you will be able to hear beeps. Press “+” or “-” you can turn up or down. Press “ESC” to exit.



2.6.9 I2C-EEPROM Test

Type “i2c –w” in a terminal you will be able to write data (0x00-0xff) to 24C08.

```
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]# i2c -w
Open /dev/i2c/0 with 8bit mode
  Writing 0x00-0xff into 24C08

 0000|   00 01 02 03 04 05 06 07    08 09 0a 0b 0c 0d 0e 0f
 0010|   10 11 12 13 14 15 16 17    18 19 1a 1b 1c 1d 1e 1f
 0020|   20 21 22 23 24 25 26 27    28 29 2a 2b 2c 2d 2e 2f
 0030|   30 31 32 33 34 35 36 37    38 39 3a 3b 3c 3d 3e 3f
 0040|   40 41 42 43 44 45 46 47    48 49 4a 4b 4c 4d 4e 4f
 0050|   50 51 52 53 54 55 56 57    58 59 5a 5b 5c 5d 5e 5f
 0060|   60 61 62 63 64 65 66 67    68 69 6a 6b 6c 6d 6e 6f
 0070|   70 71 72 73 74 75 76 77    78 79 7a 7b 7c 7d 7e 7f
 0080|   80 81 82 83 84 85 86 87    88 89 8a 8b 8c 8d 8e 8f
 0090|   90 91 92 93 94 95 96 97    98 99 9a 9b 9c 9d 9e 9f
 00a0|   a0 a1 a2 a3 a4 a5 a6 a7    a8 a9 aa ab ac ad ae af
 00b0|   b0 b1 b2 b3 b4 b5 b6 b7    b8 b9 ba bb bc bd be bf
 00c0|   c0 c1 c2 c3 c4 c5 c6 c7    c8 c9 ca cb cc cd ce cf
 00d0|   d0 d1 d2 d3 d4 d5 d6 d7    d8 d9 da db dc dd de df
 00e0|   e0 e1 e2 e3 e4 e5 e6 e7    e8 e9 ea eb ec ed ee ef
 00f0|   f0 f1 f2 f3 f4 f5 f6 f7    f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]#
```

Type “i2c –r” in a terminal you will be able to read data from 24C08.

```
 00f0|   f0 f1 f2 f3 f4 f5 f6 f7    f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]# i2c -r
Open /dev/i2c/0 with 8bit mode
  Reading 256 bytes from 0x0

 0000|   00 01 02 03 04 05 06 07    08 09 0a 0b 0c 0d 0e 0f
 0010|   10 11 12 13 14 15 16 17    18 19 1a 1b 1c 1d 1e 1f
 0020|   20 21 22 23 24 25 26 27    28 29 2a 2b 2c 2d 2e 2f
 0030|   30 31 32 33 34 35 36 37    38 39 3a 3b 3c 3d 3e 3f
 0040|   40 41 42 43 44 45 46 47    48 49 4a 4b 4c 4d 4e 4f
 0050|   50 51 52 53 54 55 56 57    58 59 5a 5b 5c 5d 5e 5f
 0060|   60 61 62 63 64 65 66 67    68 69 6a 6b 6c 6d 6e 6f
 0070|   70 71 72 73 74 75 76 77    78 79 7a 7b 7c 7d 7e 7f
 0080|   80 81 82 83 84 85 86 87    88 89 8a 8b 8c 8d 8e 8f
 0090|   90 91 92 93 94 95 96 97    98 99 9a 9b 9c 9d 9e 9f
 00a0|   a0 a1 a2 a3 a4 a5 a6 a7    a8 a9 aa ab ac ad ae af
 00b0|   b0 b1 b2 b3 b4 b5 b6 b7    b8 b9 ba bb bc bd be bf
 00c0|   c0 c1 c2 c3 c4 c5 c6 c7    c8 c9 ca cb cc cd ce cf
 00d0|   d0 d1 d2 d3 d4 d5 d6 d7    d8 d9 da db dc dd de df
 00e0|   e0 e1 e2 e3 e4 e5 e6 e7    e8 e9 ea eb ec ed ee ef
 00f0|   f0 f1 f2 f3 f4 f5 f6 f7    f8 f9 fa fb fc fd fe ff

[root@FriendlyARM /]#
```

2.6.10 AD Conversion Test

Type “adc-test” in a terminal, you will be able to test AD conversion. By adjusting the W1 resistor you will observe the output.

2.6.11 Ethernet Configuration

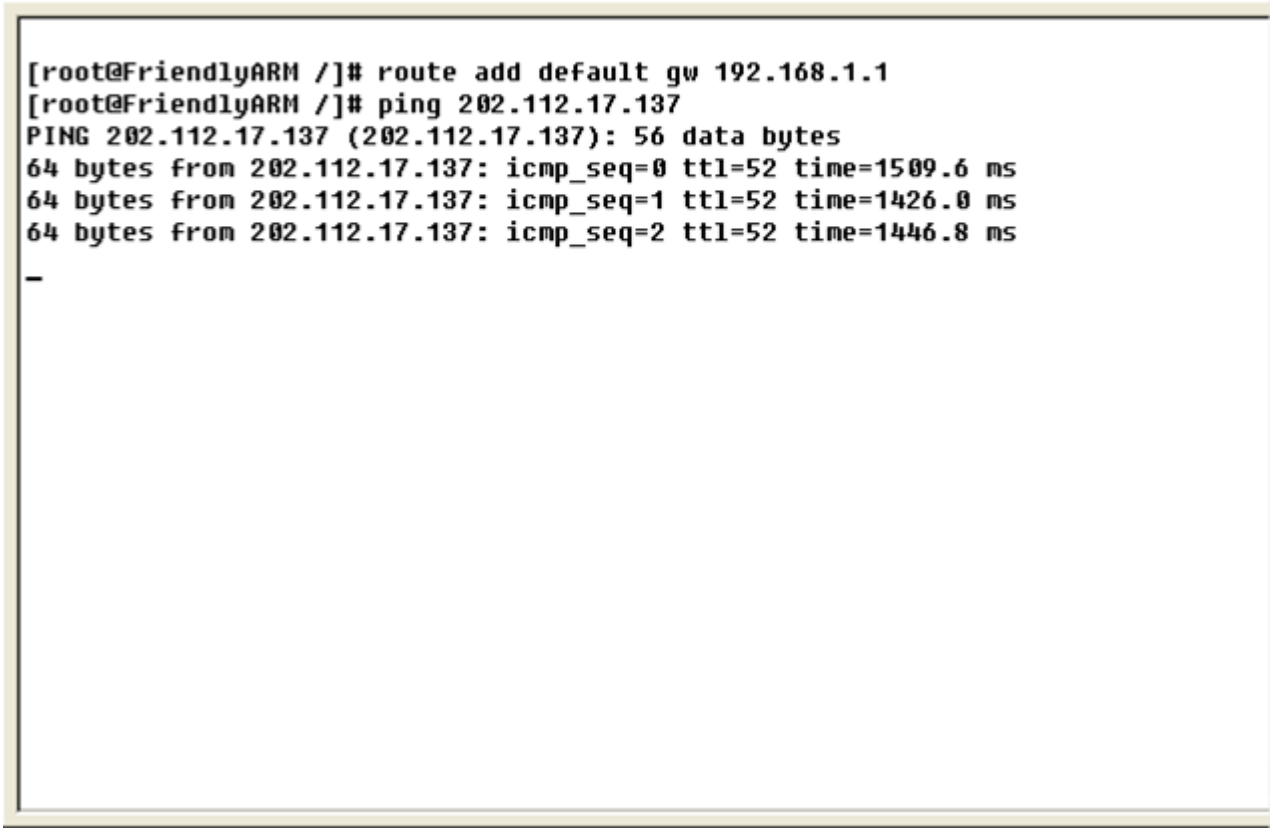
Connect your board to the internet, write down your gateway IP (the one in our example was 192.168.1.1) and configure your router:

route add default gw 192.168.1.1

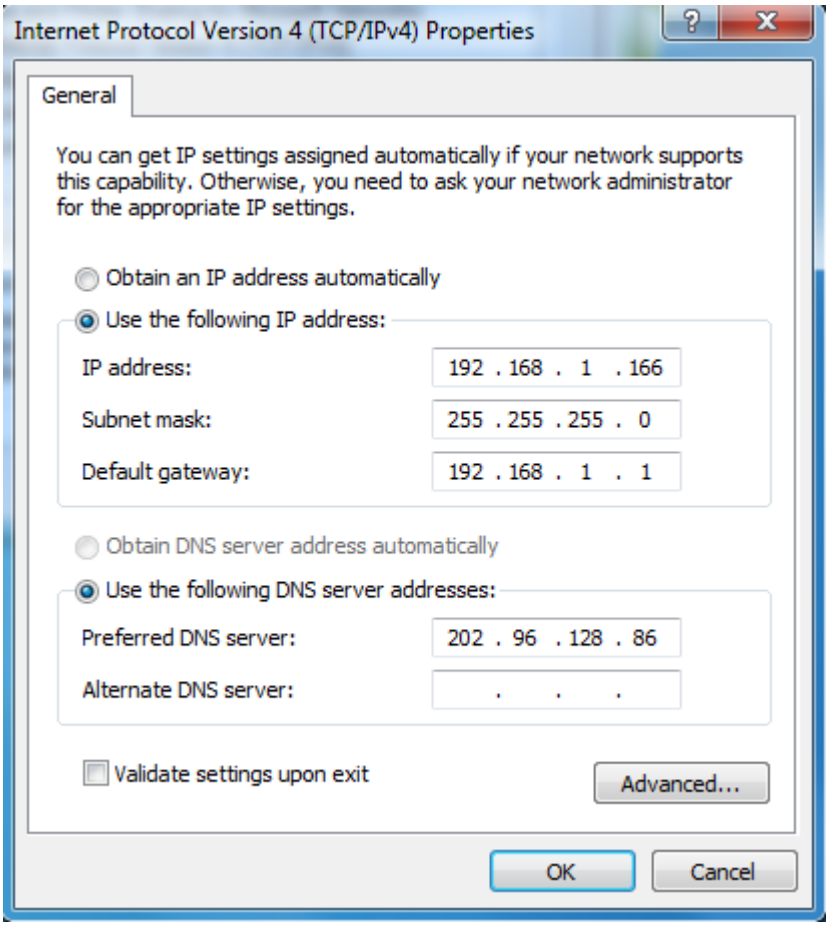
Now you can visit an IP address on the internet e.g. you can ping bbs.scut.edu.cn (IP: 202.112.17.137):

#ping 202.112.17.137

If it is a success you will see the following output



To ping through an outside website you also need to configure your DNS. You may get it from your network manager



The one in our example was “202.96.128.86”. Therefore we set our board as follows:

#rm /etc/resolv.conf; This is to remove the existing configuration file.

#touch /etc/resolv.conf; This is to generate a resolv.conf file

#echo nameserver 202.96.128.86 >> /etc/resolv.conf; Set up the DNS configuration file resolv.conf with your DNS IP or you can edit it with vi.

```
[root@FriendlyARM /]# rm /etc/resolv.conf
[root@FriendlyARM /]# touch /etc/resolv.conf
[root@FriendlyARM /]# echo nameserver 202.96.128.86 >> /etc/resolv.conf
[root@FriendlyARM /]# cat /etc/resolv.conf
nameserver 202.96.128.86
[root@FriendlyARM /]# ping www.163.com
PING www.cache.split.netease.com (220.181.28.54): 56 data bytes
64 bytes from 220.181.28.54: icmp_seq=0 ttl=53 time=1353.8 ms
64 bytes from 220.181.28.54: icmp_seq=1 ttl=53 time=1378.0 ms
64 bytes from 220.181.28.54: icmp_seq=2 ttl=53 time=1398.1 ms
64 bytes from 220.181.28.54: icmp_seq=4 ttl=53 time=1356.0 ms
64 bytes from 220.181.28.54: icmp_seq=5 ttl=53 time=1314.9 ms

--- www.cache.split.netease.com ping statistics ---
7 packets transmitted, 5 packets received, 28% packet loss
round-trip min/avg/max = 1314.9/1360.1/1398.1 ms
[root@FriendlyARM /]# _
```

2.6.12 Configure MAC Address

The MAC address in the Mini2451 is “soft” therefore you can change it via “ifconfig”.

First check your current MAC address via “ifconfig”:

#ifconfig ;

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
[root@FriendlyARM /]# cat /etc/resolv.conf
nameserver 192.168.1.1
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:90:90:90:90:90
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5236 (5.1 KiB)  TX bytes:977 (977.0 B)
          Interrupt:51

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[root@FriendlyARM /]# _
```

In our example the MAC was “08：90：90：90：90：90”, this is the default MAC address and has been hard-coded in the kernel. If you want to update it you have to recompile the kernel. In order to change the MAC dynamically you need to close your network connection and then fill your new MAC:

#ifconfig eth0 down

#ifconfig eth0 hw ether 00:11:AA:BB:CC:DD; note: **a,b,c,d,e,f…** could be lower case

Restart the network, check your MAC via “ifconfig” and verify your network via “ping”:

#ifconfig eth0 up

#ifconfig

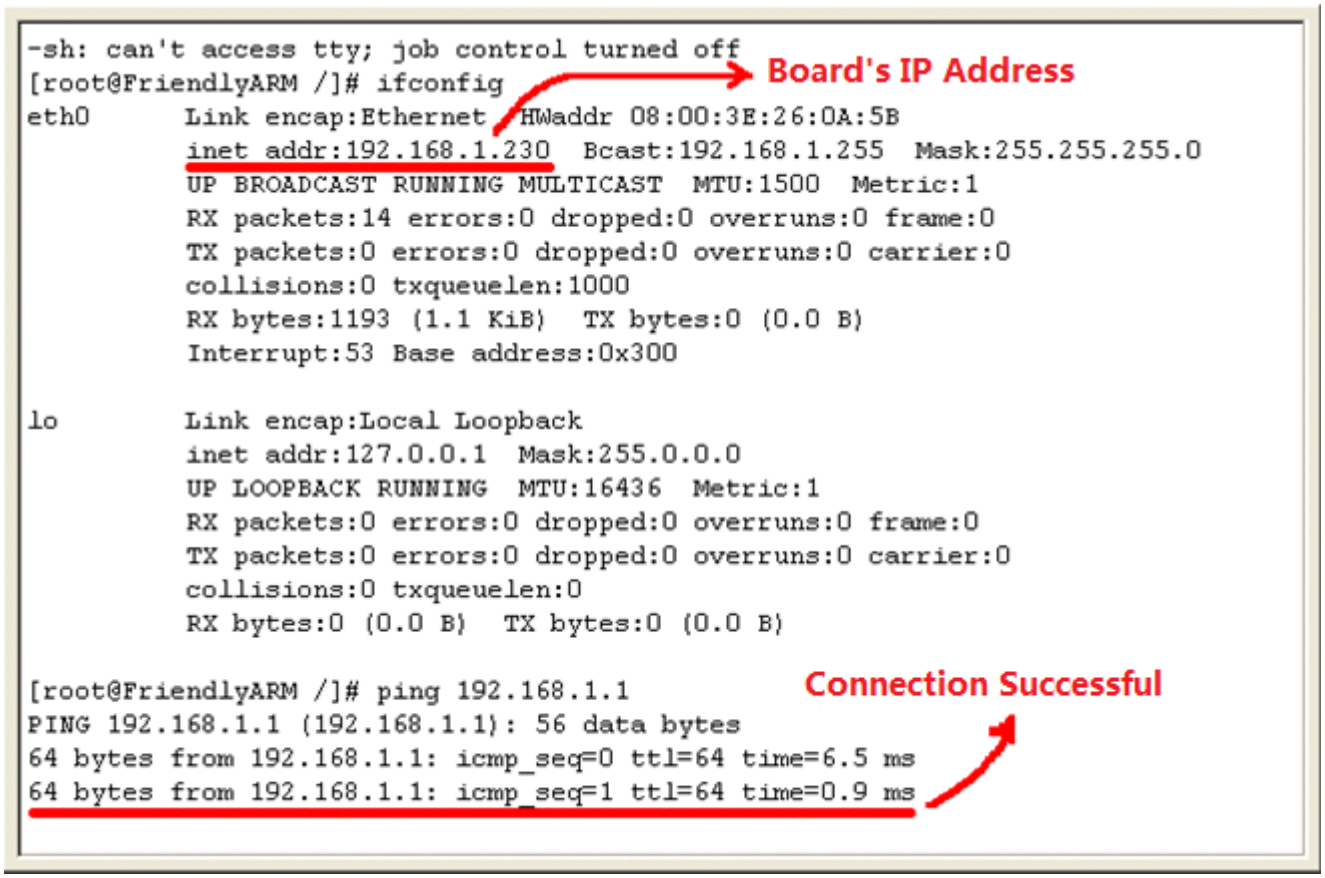
#ping 192.168.1.1

```
[root@FriendlyARM /]# ifconfig eth0 hw ether 00:11:aa:bb:cc:dd
[root@FriendlyARM /]# ifconfig eth0 up
[root@FriendlyARM /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:AA:BB:CC:DD
          inet addr:192.168.1.230  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4791 (4.6 KiB)  TX bytes:672 (672.0 B)
          Interrupt:53 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[root@FriendlyARM /]# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=3.2 ms
```

2.6.13 Telnet

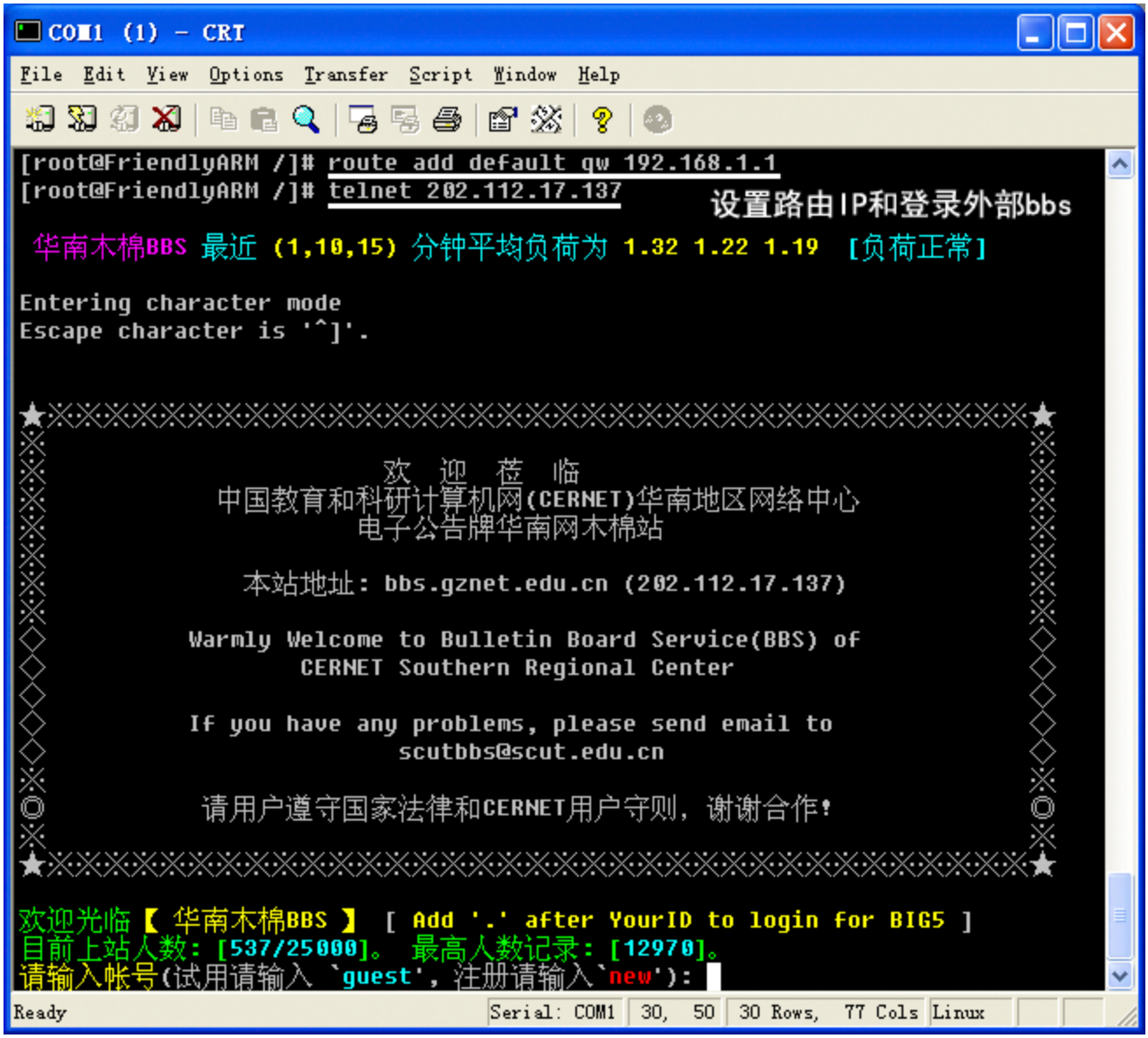
"Telnet" is a popular utility. If your board is connected to the internet you can telnet a bbs.

First make sure your board's IP is 192.168.1.230 and your board is communicating with other machines.



Then configure your router's IP: **route add default gw 192.168.1.1**

Now you can telnet a BBS. Here we visited "bbs.scut.edu.cn".

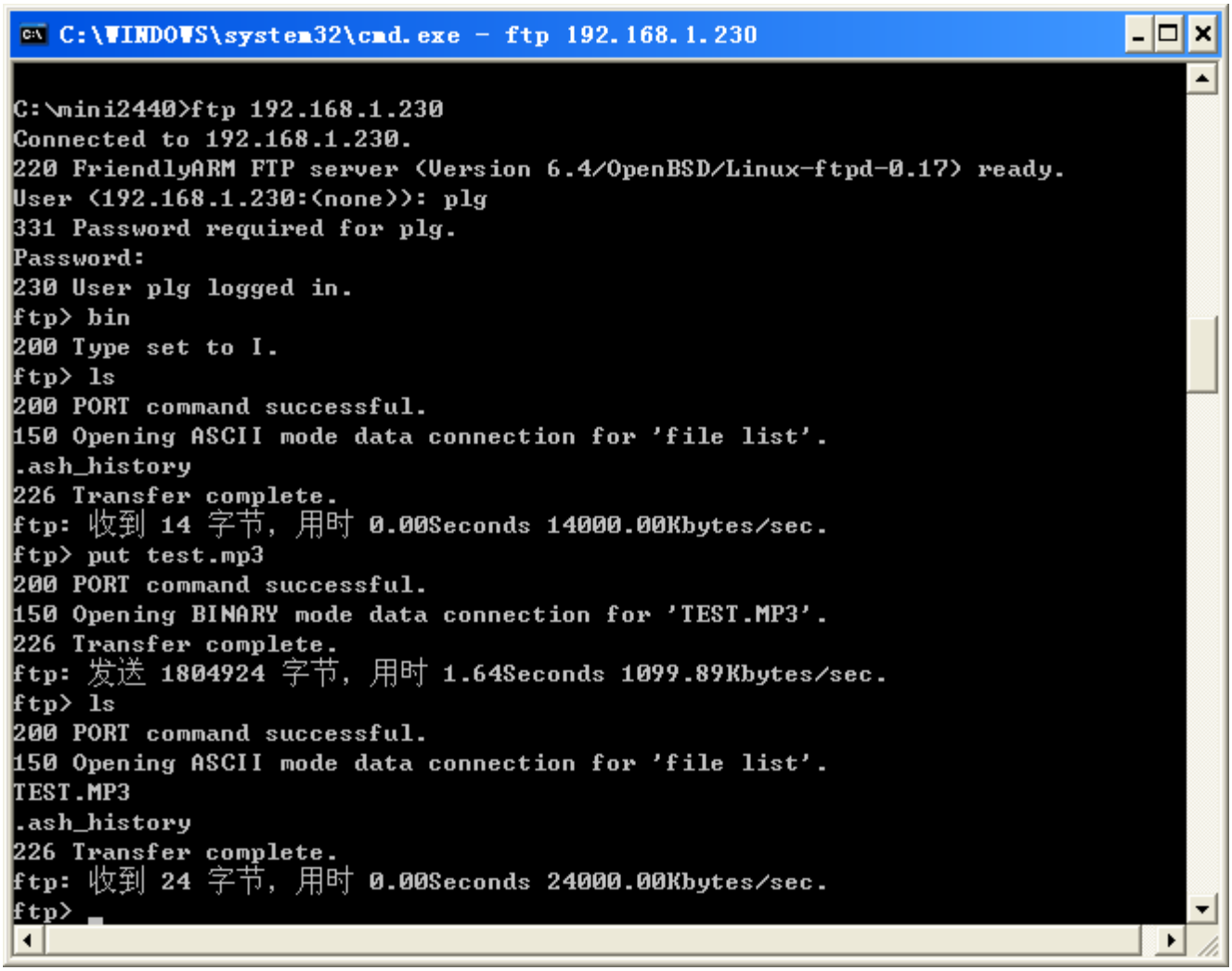


2.6.14 File Transfer with FTP

After the system boots normally it will automatically start a telnet service. Users can ftp a remote host via “ftp” in the command line utility in both Linux and Windows. Users can transfer files to the board from a host PC.

Note: please make sure you have a file ready in your FTP directory. Here we had “test.mp3”.The account for login is plg and the password is plg.

After file transfer is doen you will see a test.mp3 file in your target board's /home/plg directory.



2.6.15 Control LED Remotely

Click on "Manipulating LEDs via HTML" on the test page of our web server the following page will be loaded

You can test each of these items. The “LED Test” manipulates the LEDs via CGI and it includes two display modes and three display rates.

To stop the web service you need to type the following commands:

#/etc/rc.d/init.d/httpd stop

Then restart the service

#/etc/rc.d/init.d/httpd start

2.6.16 Mount NFS

Please make sure you have set up the NFS server in your host PC and then type the

following command (our server's IP is 192.168.1.111).

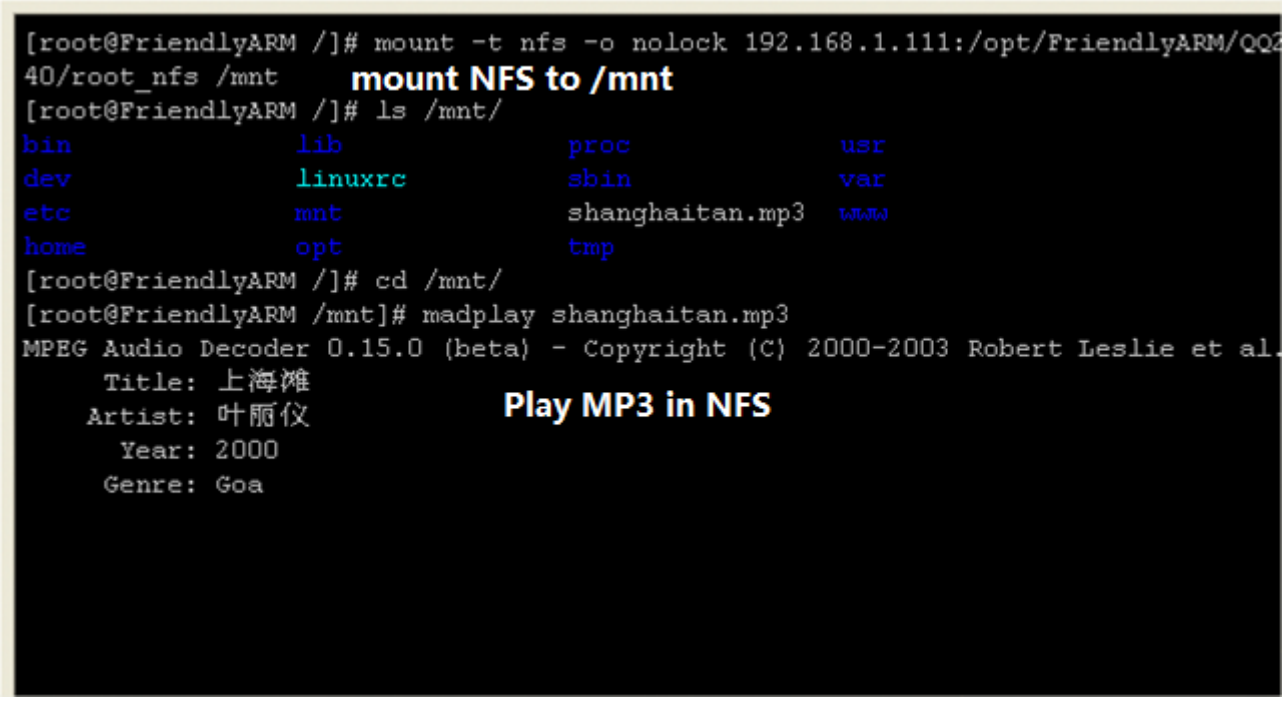
#mount –t nfs –o nolock

192.168.1.111 :/opt/FriendlyARM/mini2451/rootfs_qtopia /mnt

After a successful mount you will be able to enter "/mnt" and operate your files

To unmount it type the command below

#**umount /mnt**



2.6.17 Set System Clock

The Linux command for updating time is "**date**", to synchronize the S3C2451 time with Linux's system time you can use "**hwclock**":

(1) date -s 042916352007 #set time to 2007-04-29 16:34

(2) hwclock -w # save your setting to S3C2451's RTC

(3) Command "hwclock –s" to update Linux's system time with RTC. Usually this

command will be included in "/etc/init.d/rcS" for auto run

Note: our system's "/etc/init.d/rcS" includes "*hwclock –s*" already.

2.6.18 Save Data to Flash Permanently

The Mini2451 system supports yaffs2 thus can save data permenantly even when the system is powered off. After the system boots please try the following command:

#cp / shanghaitan.mp3 /home/plg

This will create a duplicate file under "/home/plg". Power off and on you will observe that the file still exists.

2.6.19 Setup Autorun Program on System Startup

Users can set up programs that will be automatically run on system startup in the boot script. It is similar to Window's Autobat. It is under the /etc/init.d/rcS directory, the contents are as follows (they may be different in differed systems)

```
#! /bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:
runlevel=S
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
/bin/hostname FriendlyARM
[ -e /proc/1 ] || /bin/mount -n -t proc none /proc
[ -e /sys/class ] || /bin/mount -n -t sysfs none /sys
[ -e /dev/tty ] || /bin/mount -t ramfs none /dev
/bin/mount -n -t usbfs none /proc/bus/usb
echo /sbin/mdev > /proc/sys/kernel/hotplug
```

```
/sbin/mdev -s
/bin/hotplug
# mounting file system specified in /etc/fstab
mkdir -p /dev/pts
mkdir -p /dev/shm
/bin/mount -n -t devpts none /dev/pts -o mode=0622
/bin/mount -n -t tmpfs tmpfs /dev/shm
/bin/mount -n -t ramfs none /tmp
/bin/mount -n -t ramfs none /var
mkdir -p /var/empty
mkdir -p /var/log
mkdir -p /var/lock
mkdir -p /var/run
mkdir -p /var/tmp
/sbin/hwclock -s
syslogd
/etc/rc.d/init.d/netd start
echo " " > /dev/tty1
echo "Starting networking..." > /dev/tty1
sleep 1
/etc/rc.d/init.d/httpd start
echo " " > /dev/tty1
echo "Starting web server..." > /dev/tty1
sleep 1
/etc/rc.d/init.d/leds start
echo " " > /dev/tty1
echo "Starting leds service..." > /dev/tty1
echo " "
sleep 1
echo " " > /dev/tty1
/etc/rc.d/init.d/alsaconf start
echo "Loading sound card config..." > /dev/tty1
echo " "
/sbin/ifconfig lo 127.0.0.1
/etc/init.d/ifconfig-eth0
/bin/qtopia &
echo " " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
```

2.6.20 Take Screenshot with Snapshot

Users can take screenshots with “snapshot” and save them as png files

#**snapshot pic.png**

Executing this command will take a screenshot of the current LCD display and save it as “pic.png”.

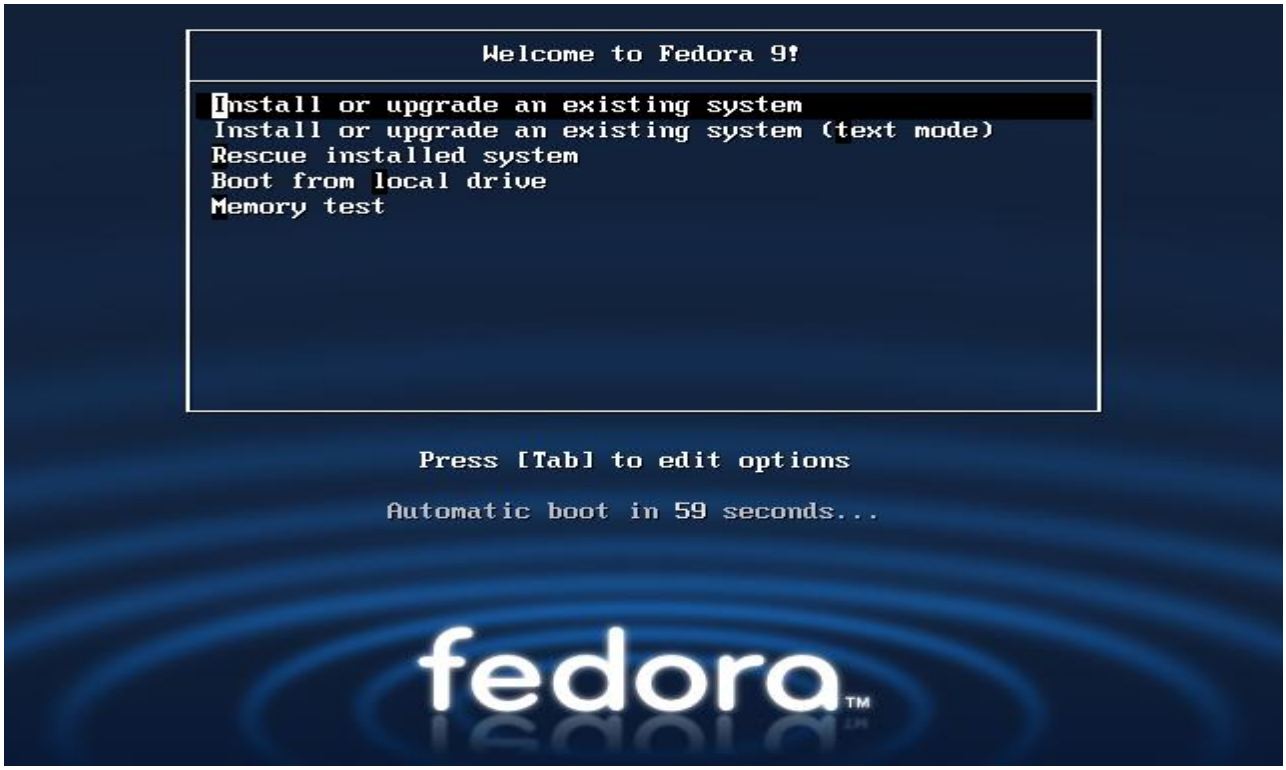
3 Set up Fedora 9.0 Development Environment

This section will guide you through the steps on how to install Fedora 9.0 on a PC and set up your Linux development environment. All our software development and testing for the Mini2451 were based on Fedora 9.0. We didn't test it on other platforms. We strongly suggest you use this platform as we do, which you can download from its website (ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso).

The reason why we chose Fedora 9.0 is that later versions such as Fedora 14/Ubuntu12 don't support development work with Qtopia2.2.0 and are more complicated and therefore may not be easy for beginners. Fedora 8 and earlier versions are a little bit obsolete. Please follow the steps below to install.

3.1 Install Fedora 9.0

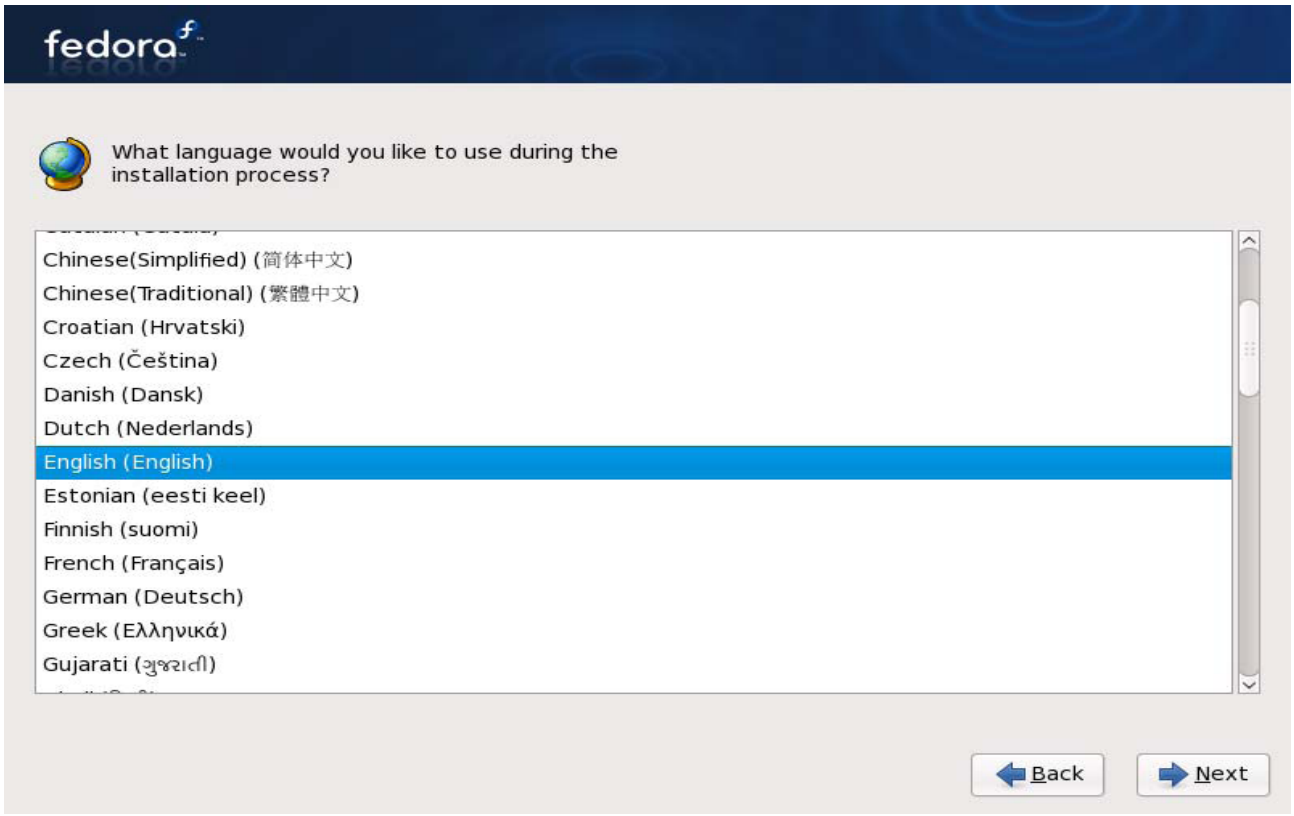
Step1： Insert the first disk in the CDROM/DVD, set the boot sequence to CDROM in the BIOS. After reboot the system, it will prompt the user to the following interface, just press "enter"



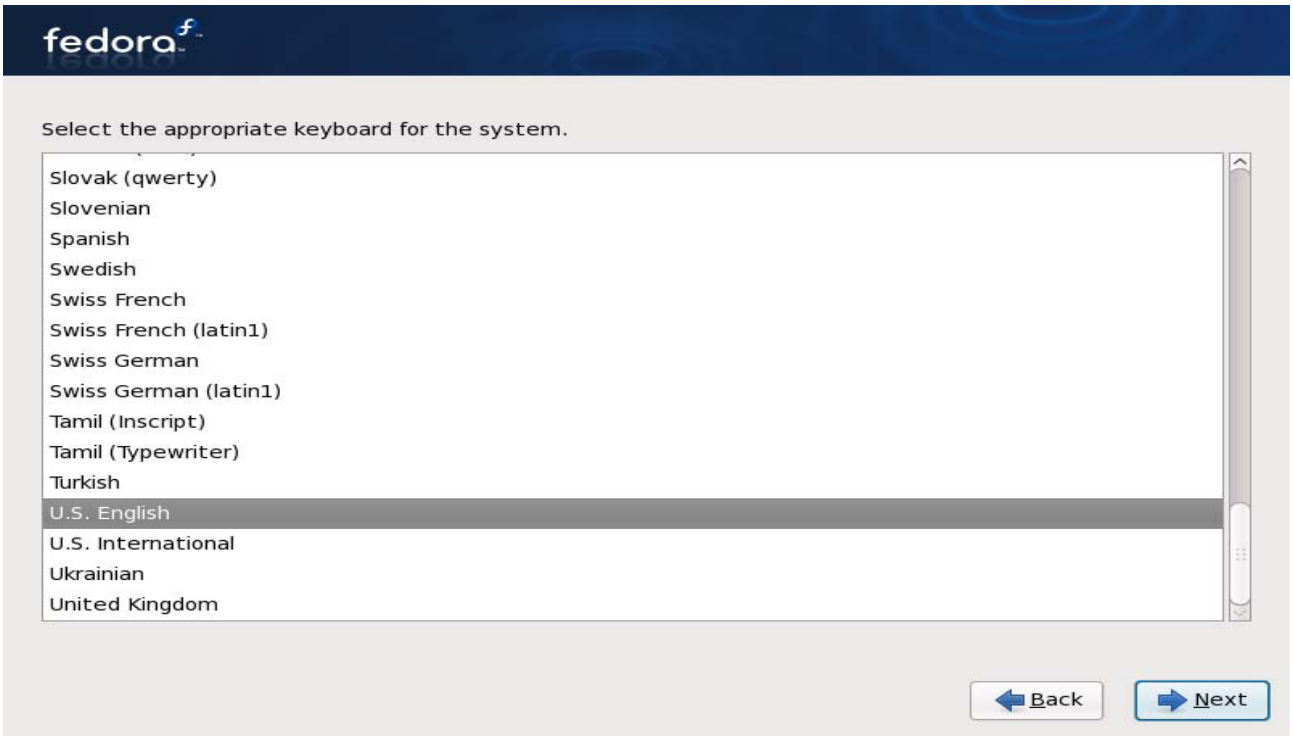
Step2：The system will check the installation disk. It can be ignored, just press “Skip” to the next step

Step3：it enters the graphic interface, click on the “Next” button.

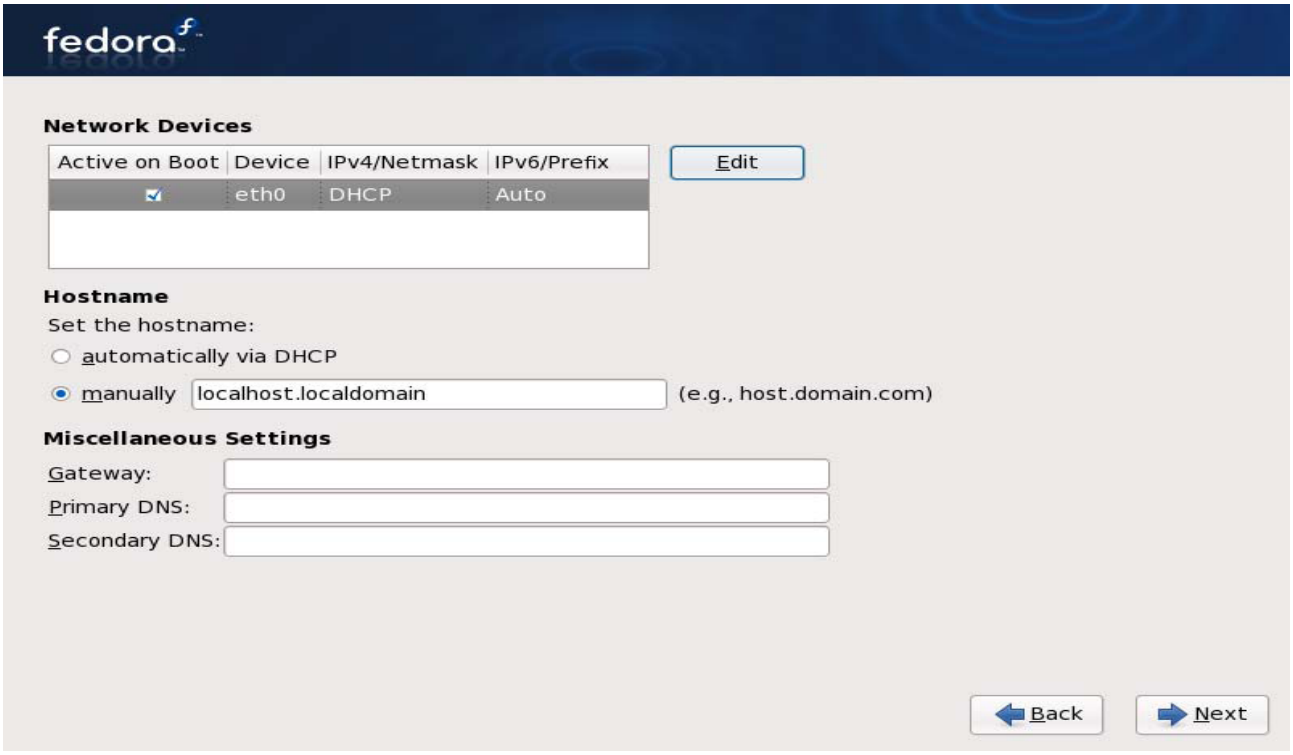
Step4：set the installation language. In this example, we chose the simplified English.



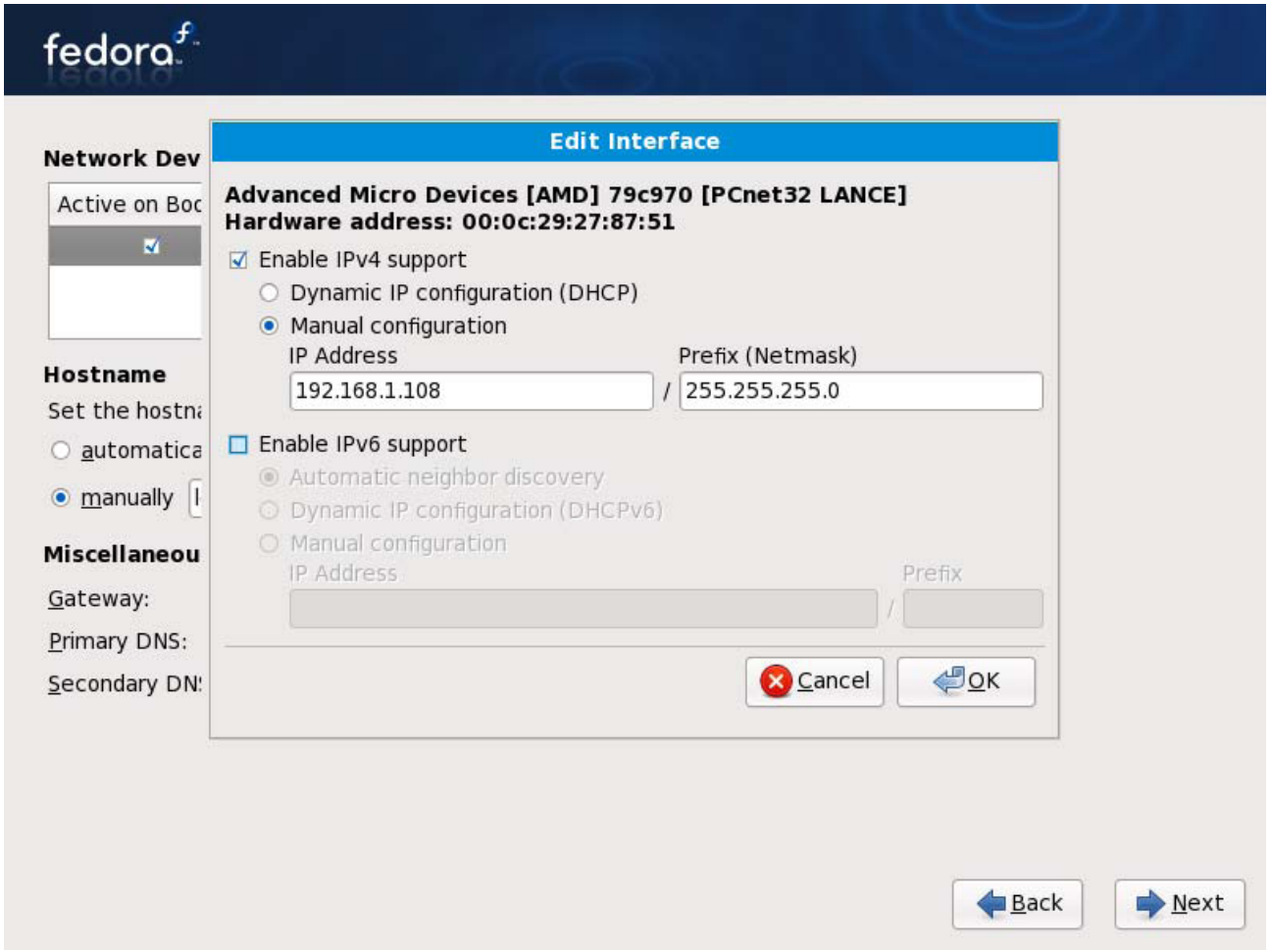
Step5：set the keyboard, in this example, we chose the U.S. key board.



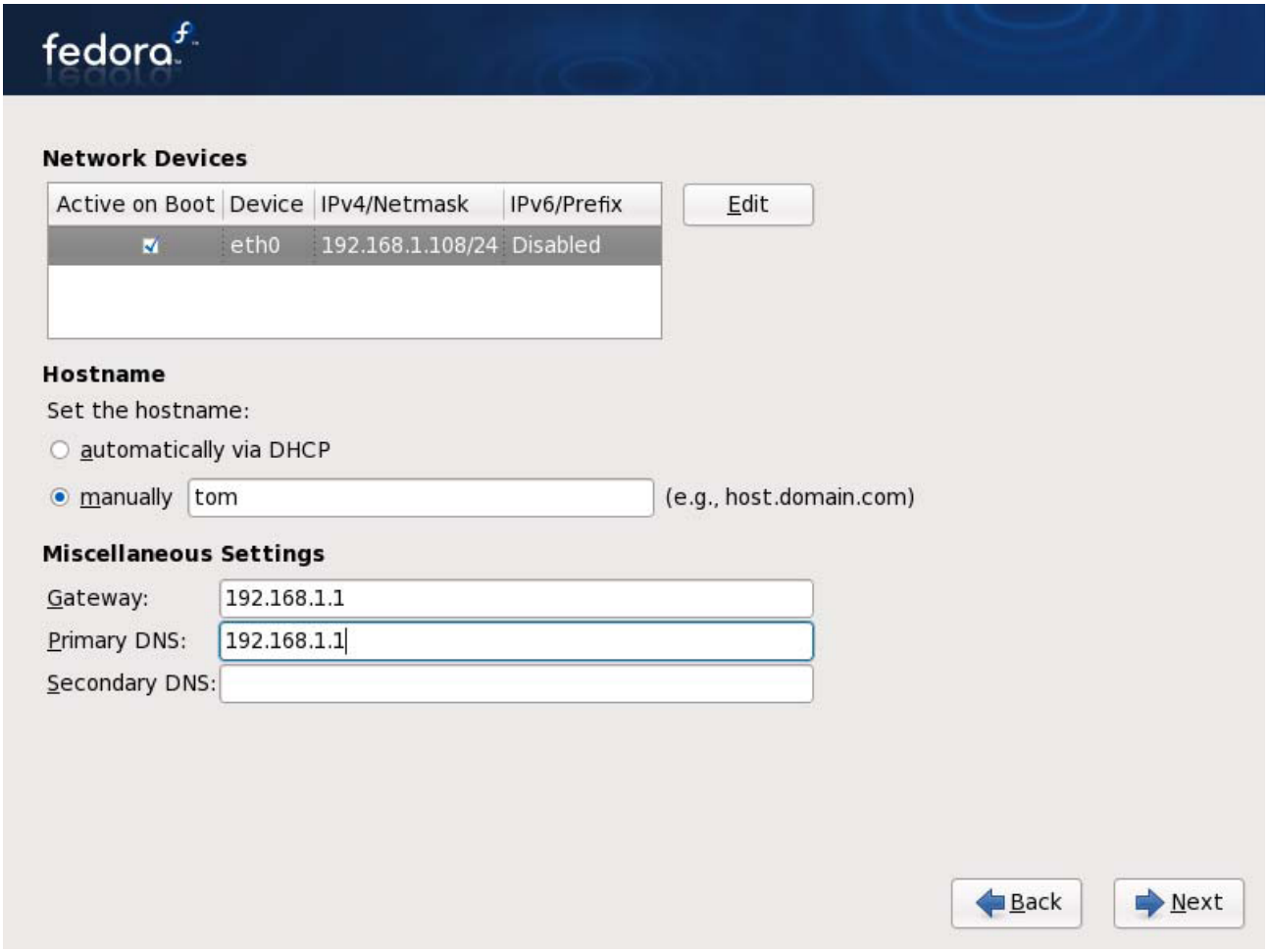
Step 6：configure the network.



In our example, we didn't set it as "DHCP", we used a static IP instead, and typed the IP and subnet mask as follows.

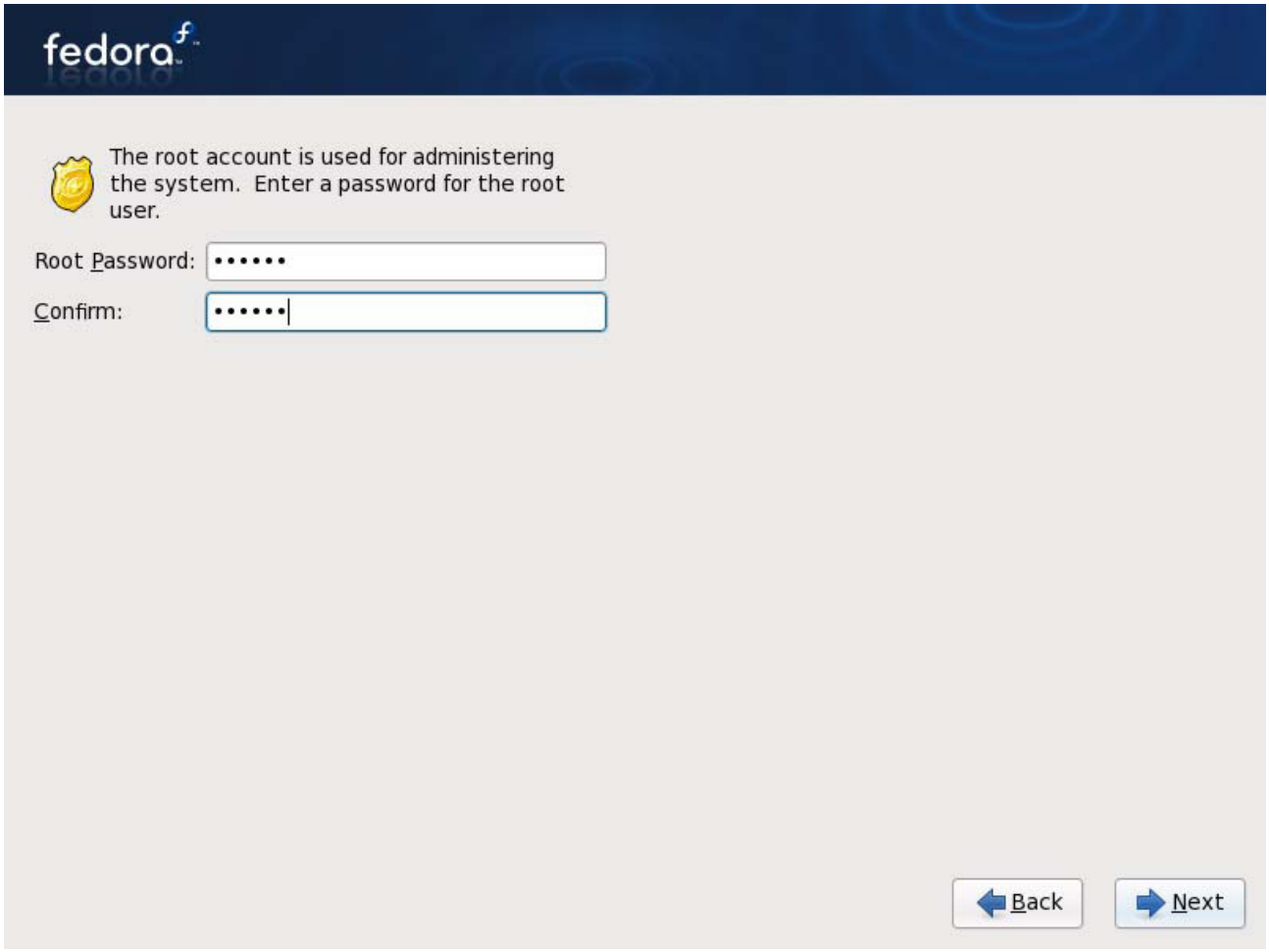


Click on the OK button and go on to set the machine name, gateway and DNS.

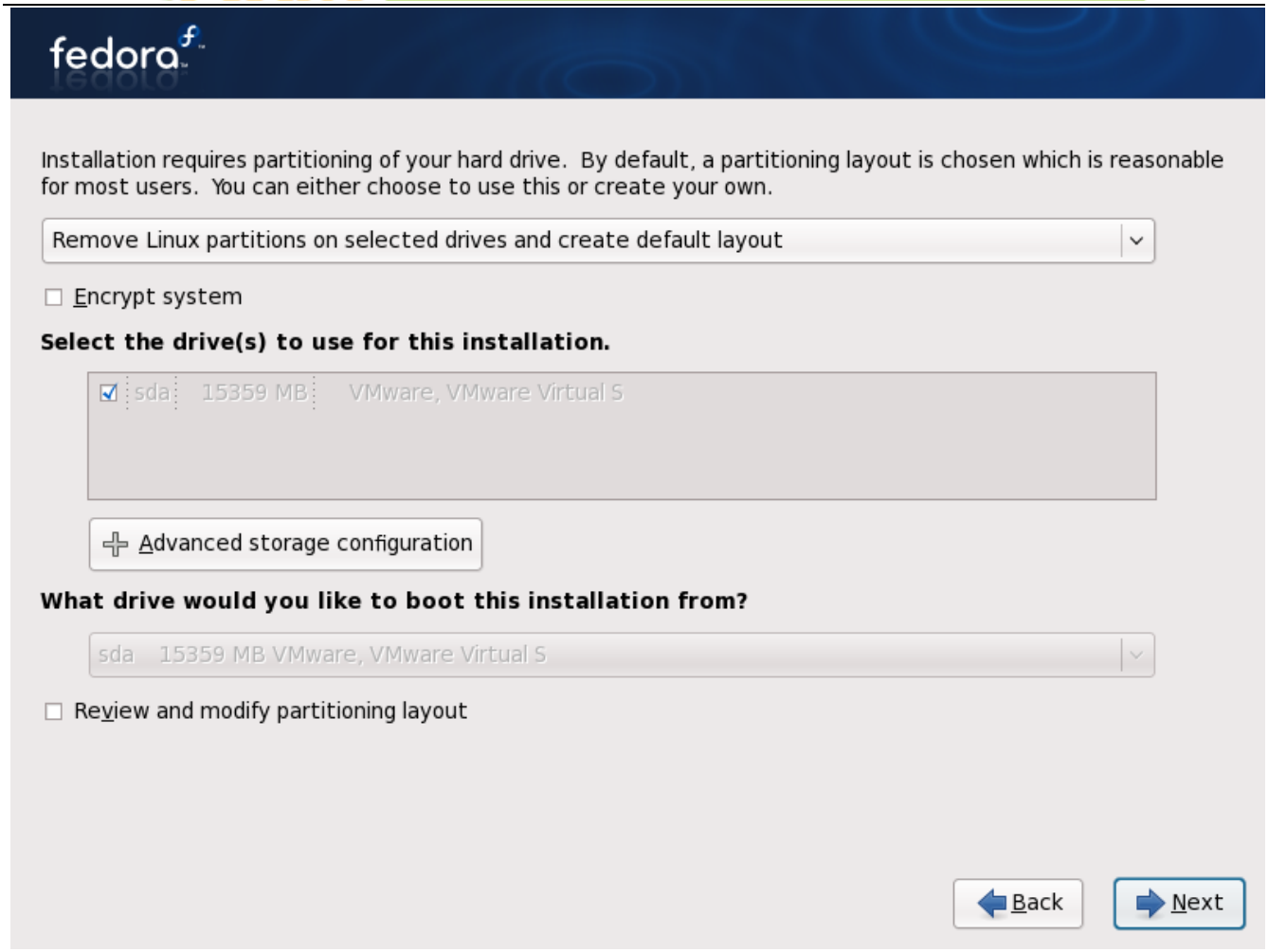


Step 7：set the time zone. We chose “Asia/Shanghai”.

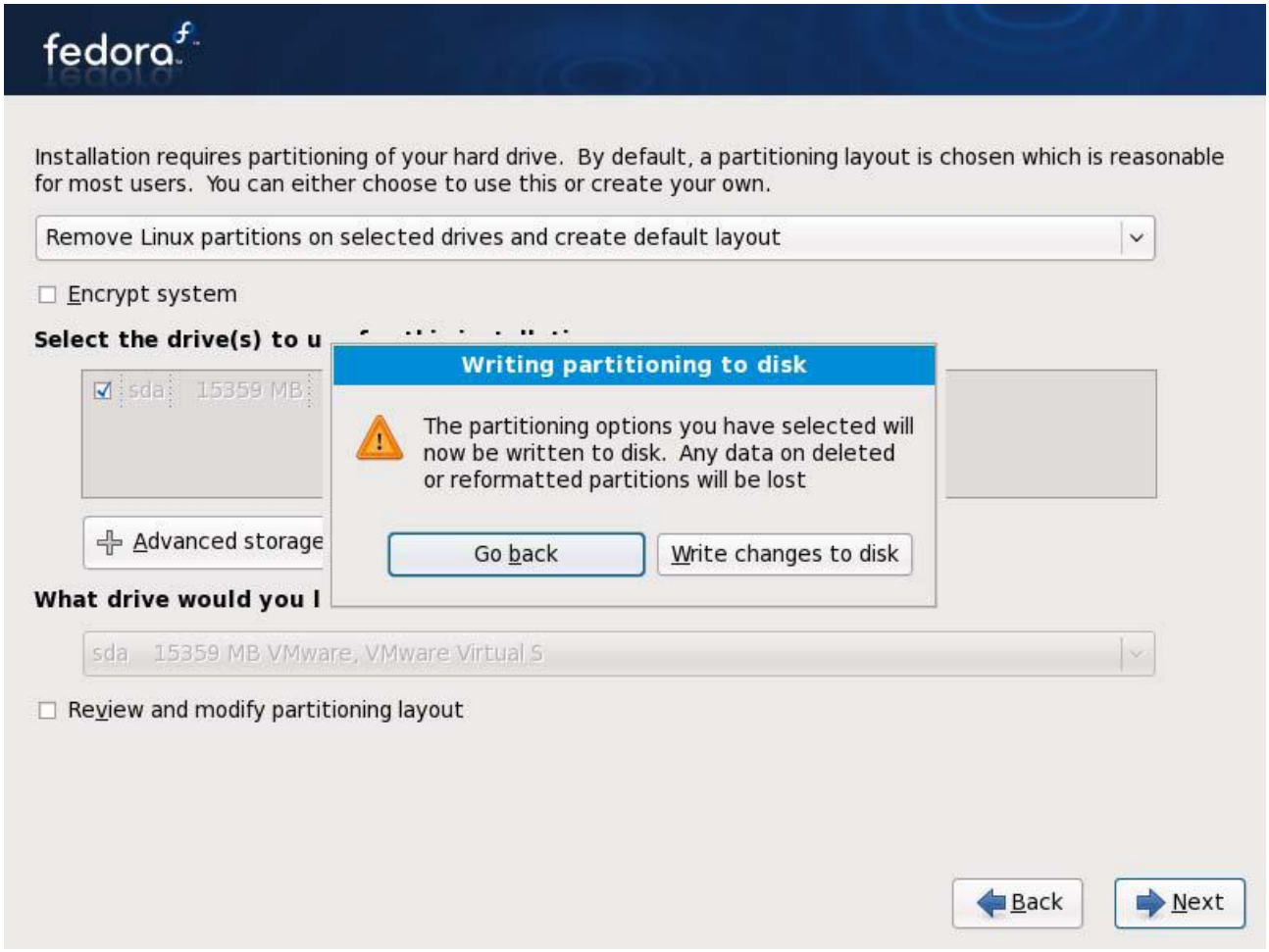
Step 8：set up the administrator’s password, i.e. the root’s password. “root” is the super user. It should be at least 6 characters



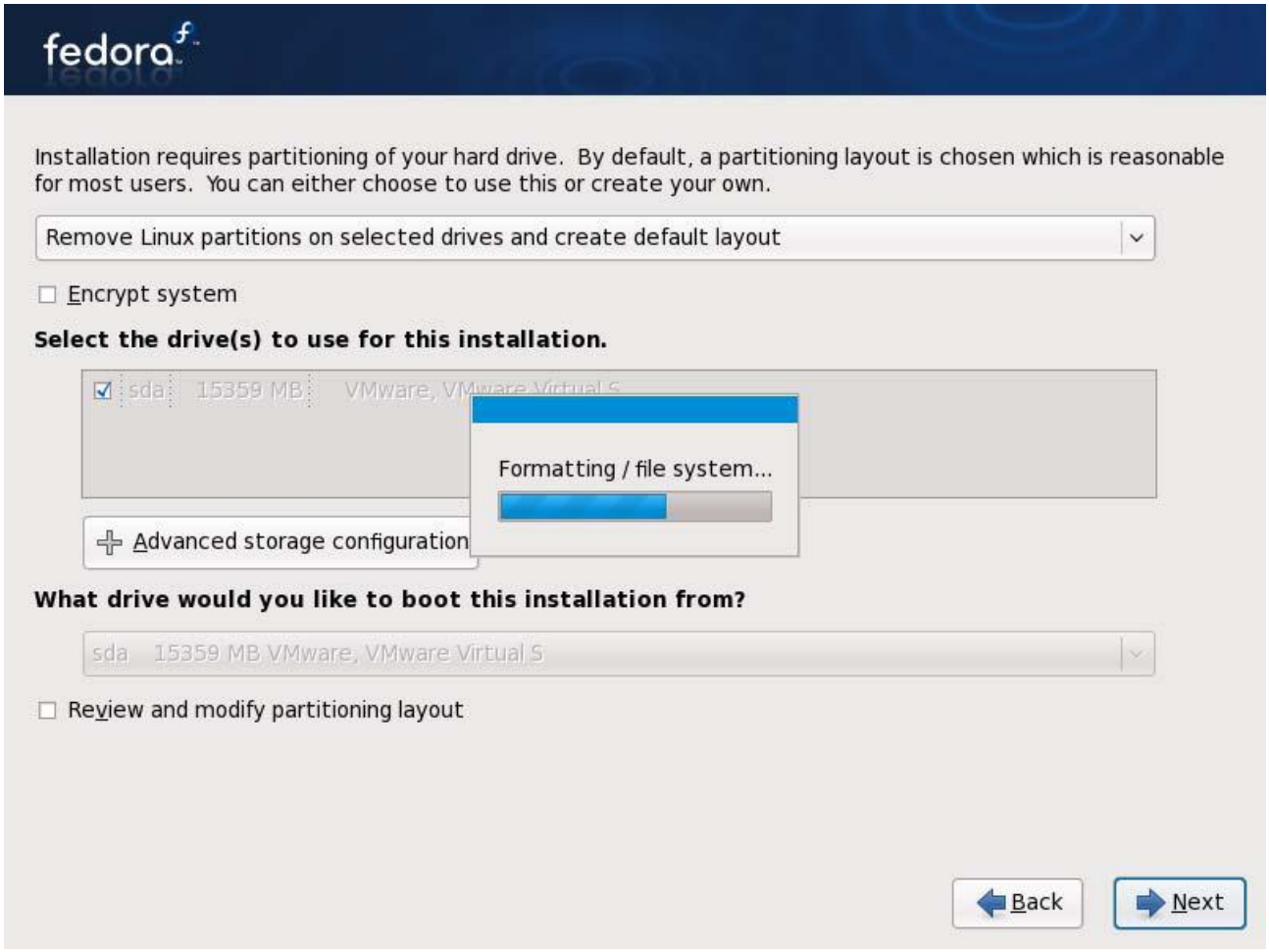
Step 9：disk partition. We followed the default option. Before do this, please back up disk data.



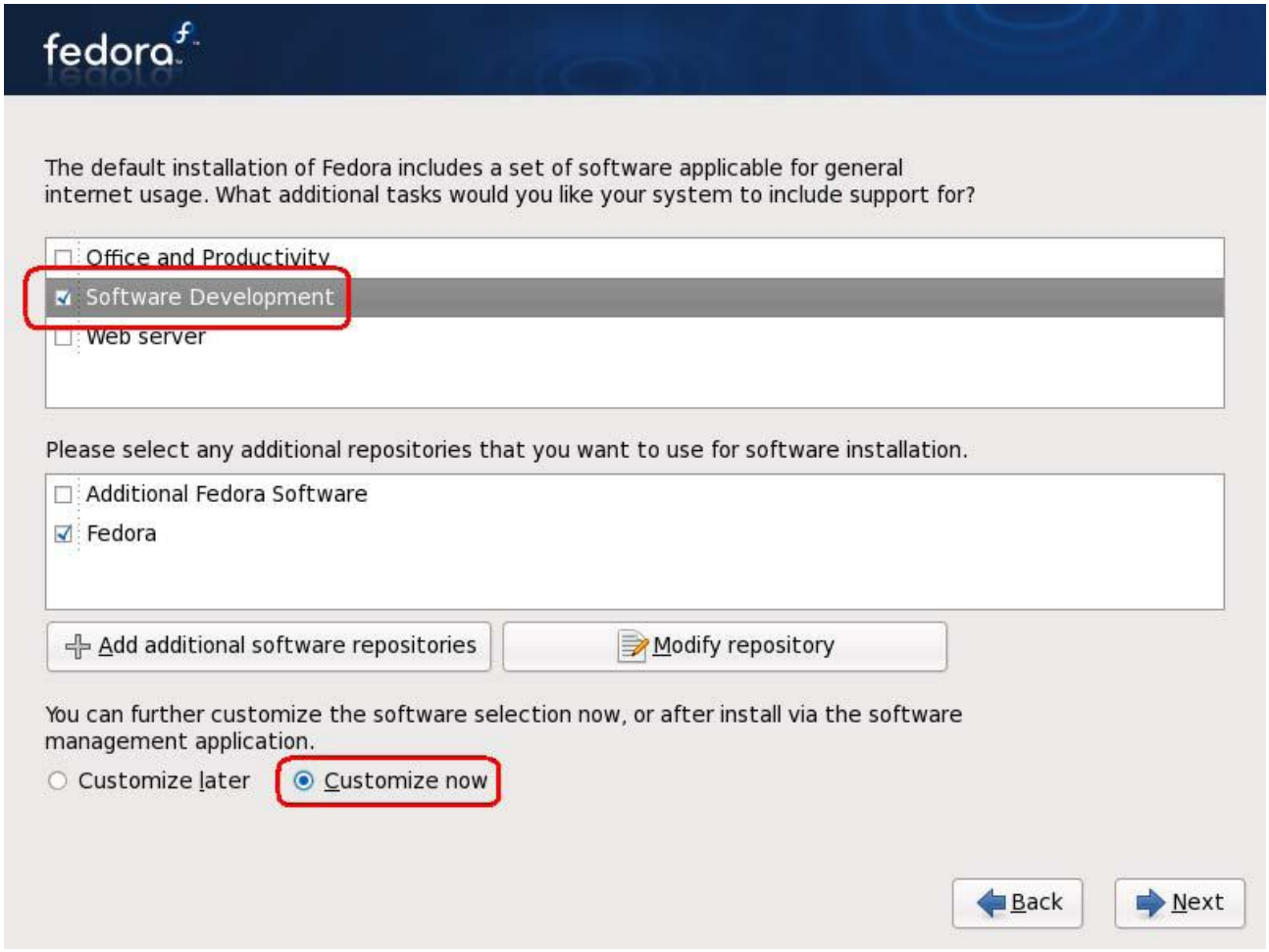
Click on “Next”, it will warn the user that all the data will be deleted. Usually we would do this installation in VMWARE, so we chose “Write changes to disk” and disk format would begin.



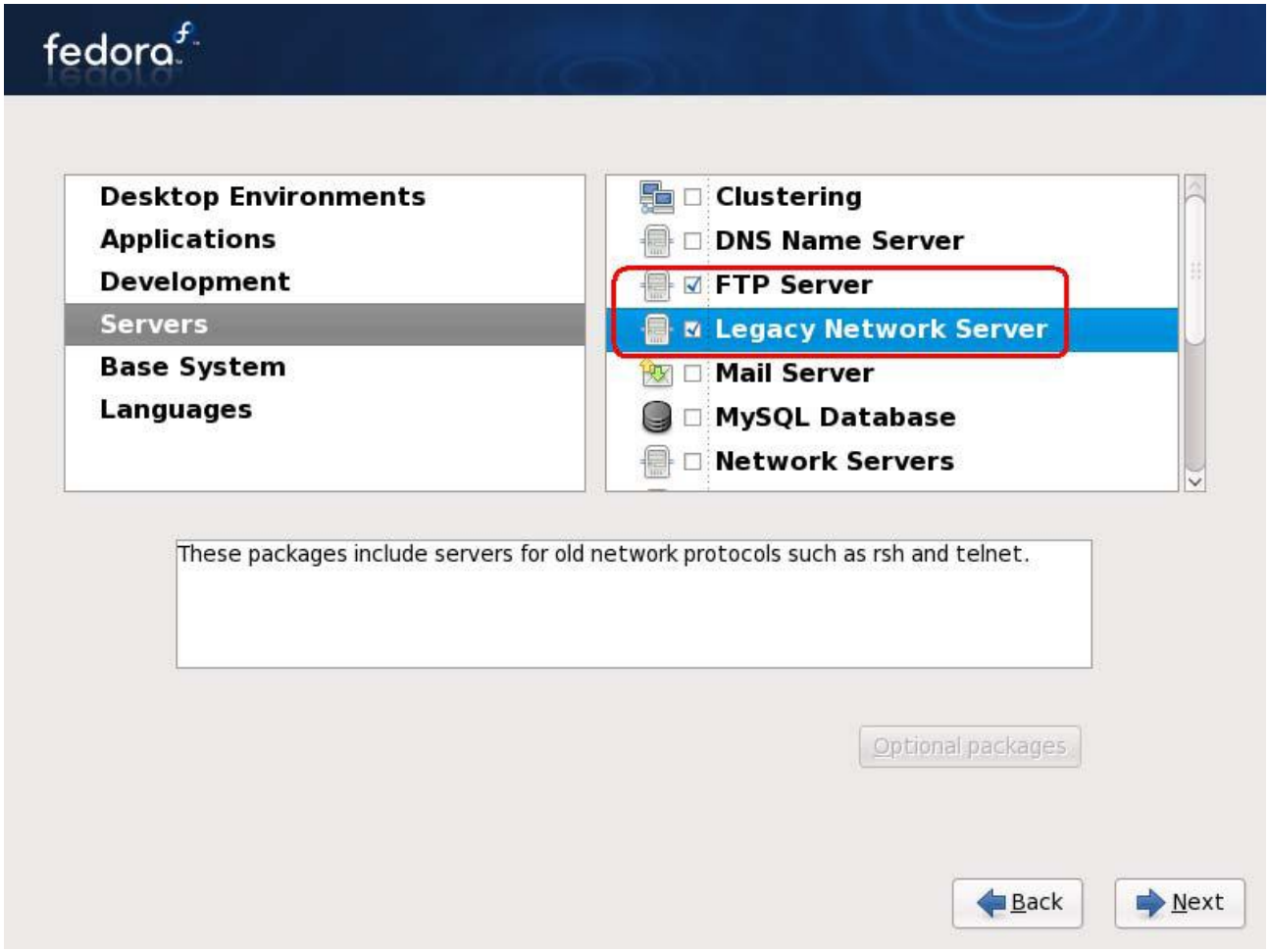
Here is the format process:



Step 11：select the installation type, in this example, we chose "customize"

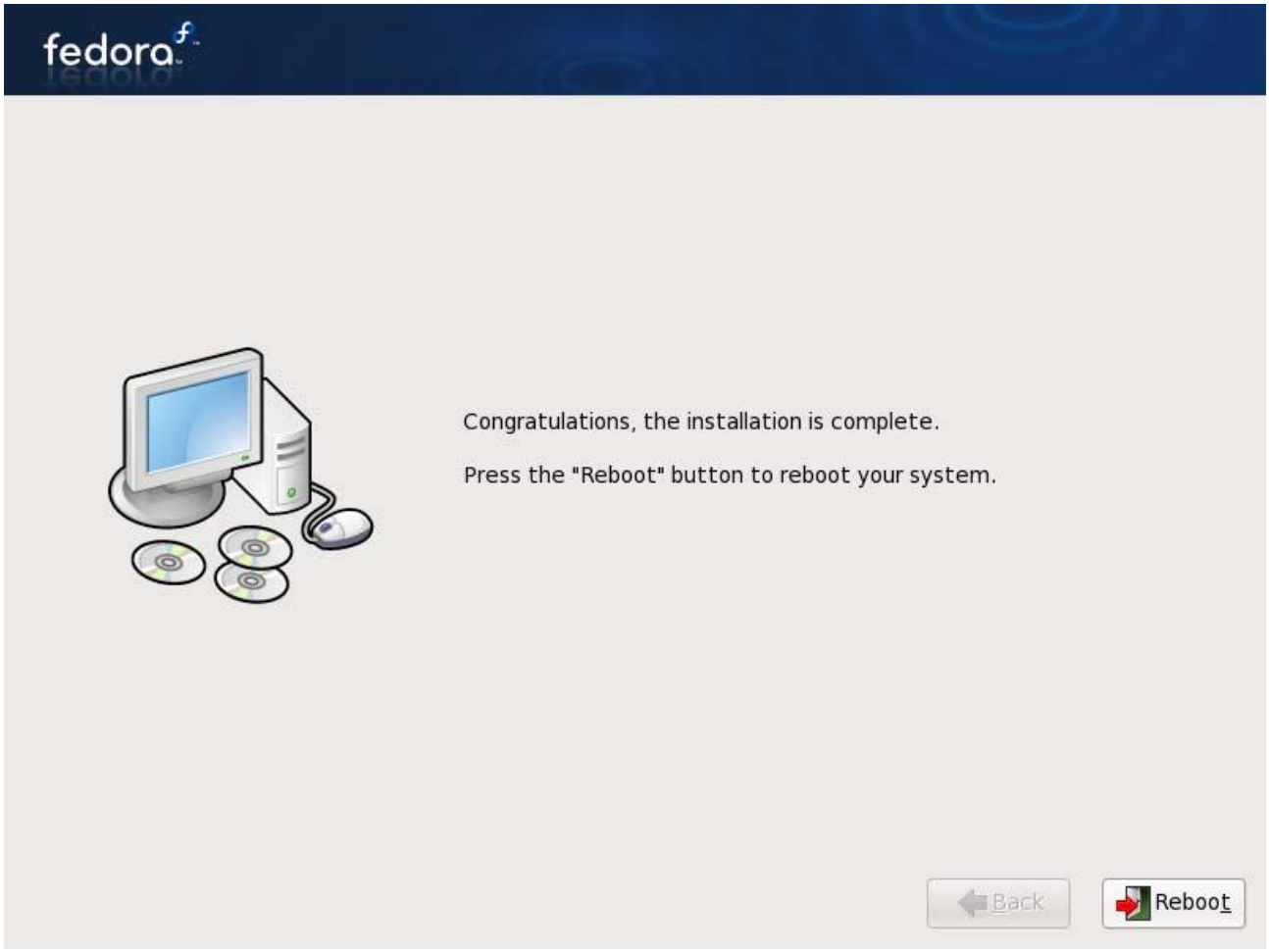


Step 12：configure the “server” item as follows:



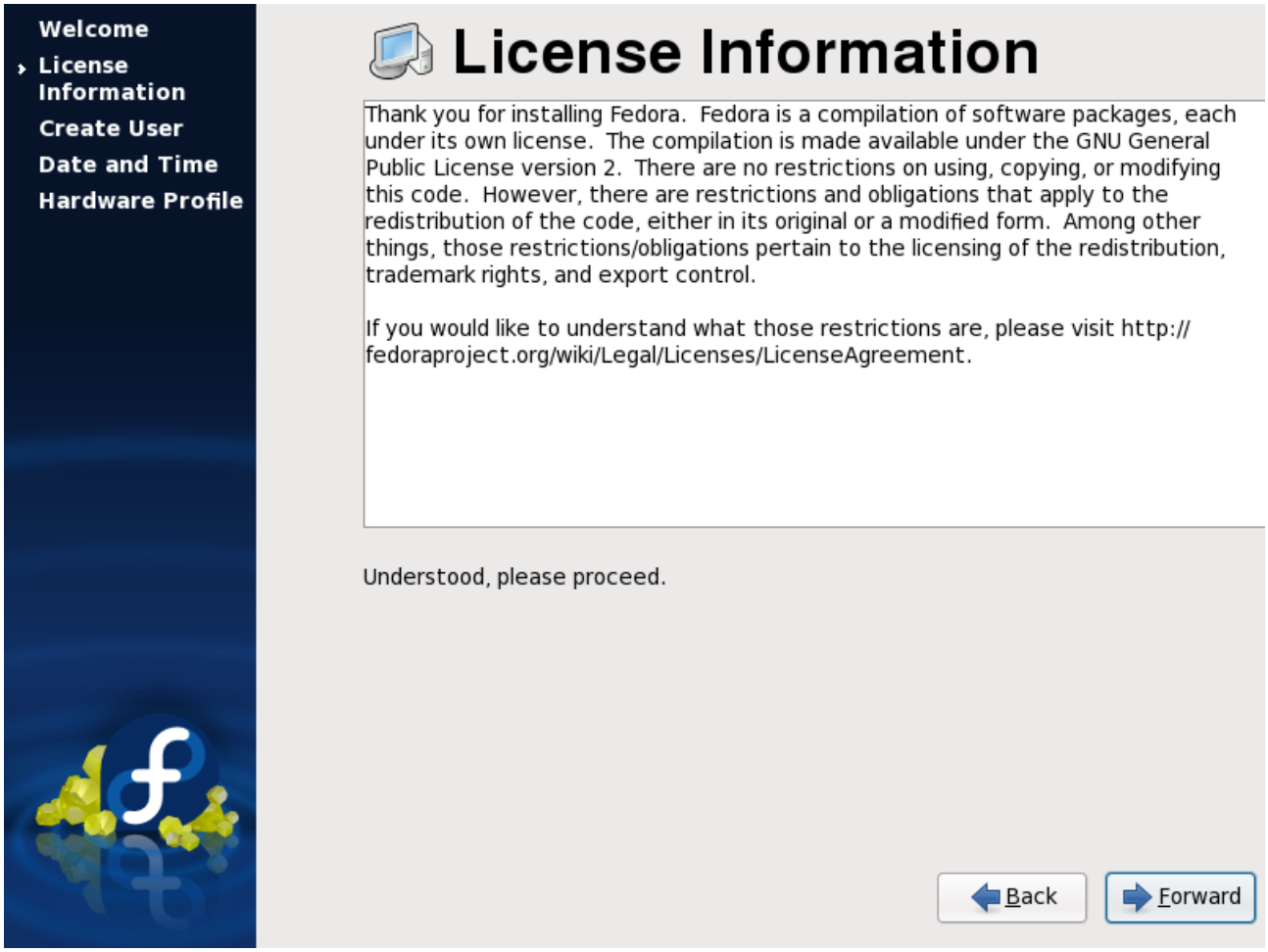
Step 13：begin installation

Step14: installation complete.

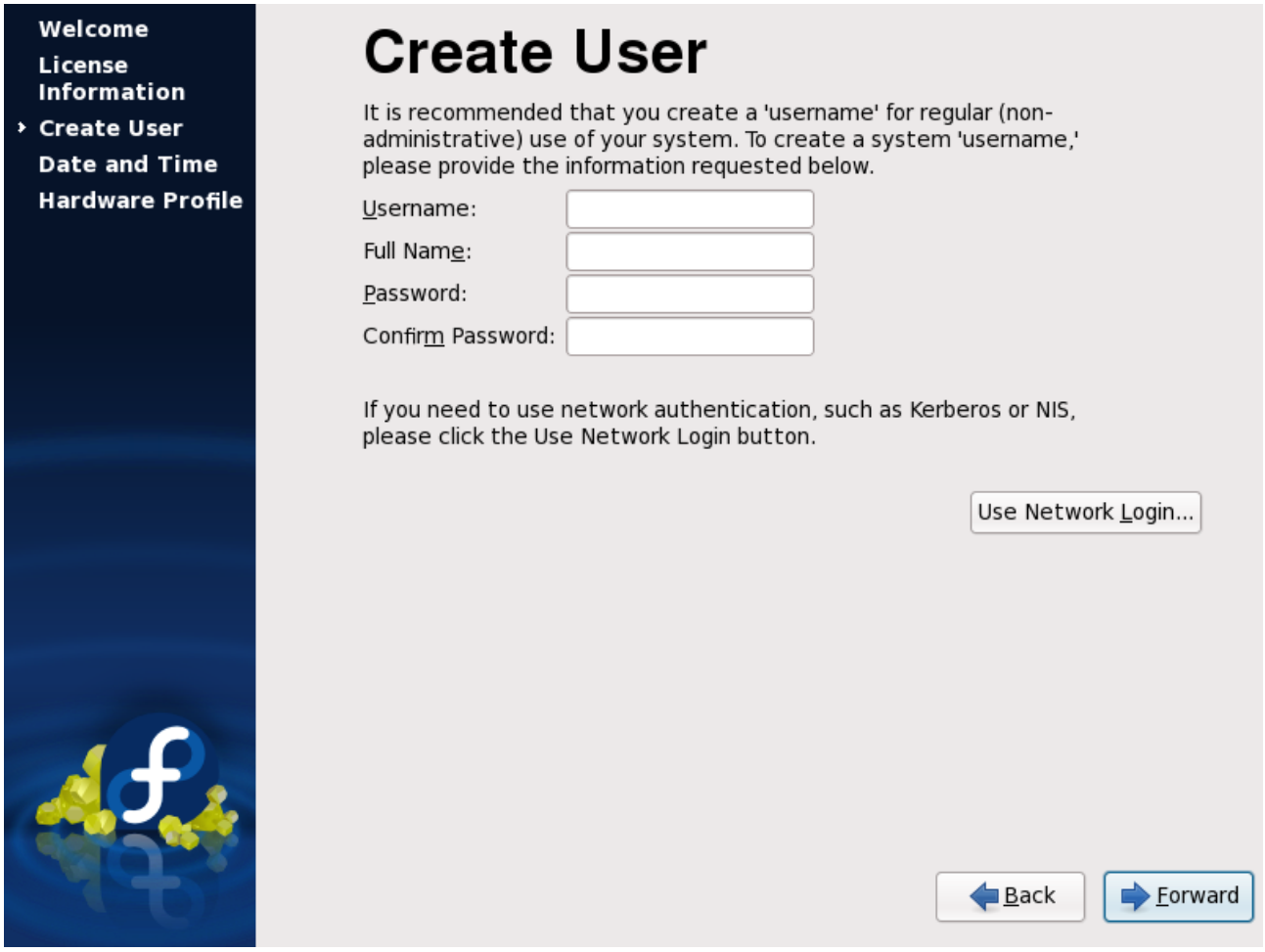


Step15: after installation completed, click on the reboot button on the page shown in step 14

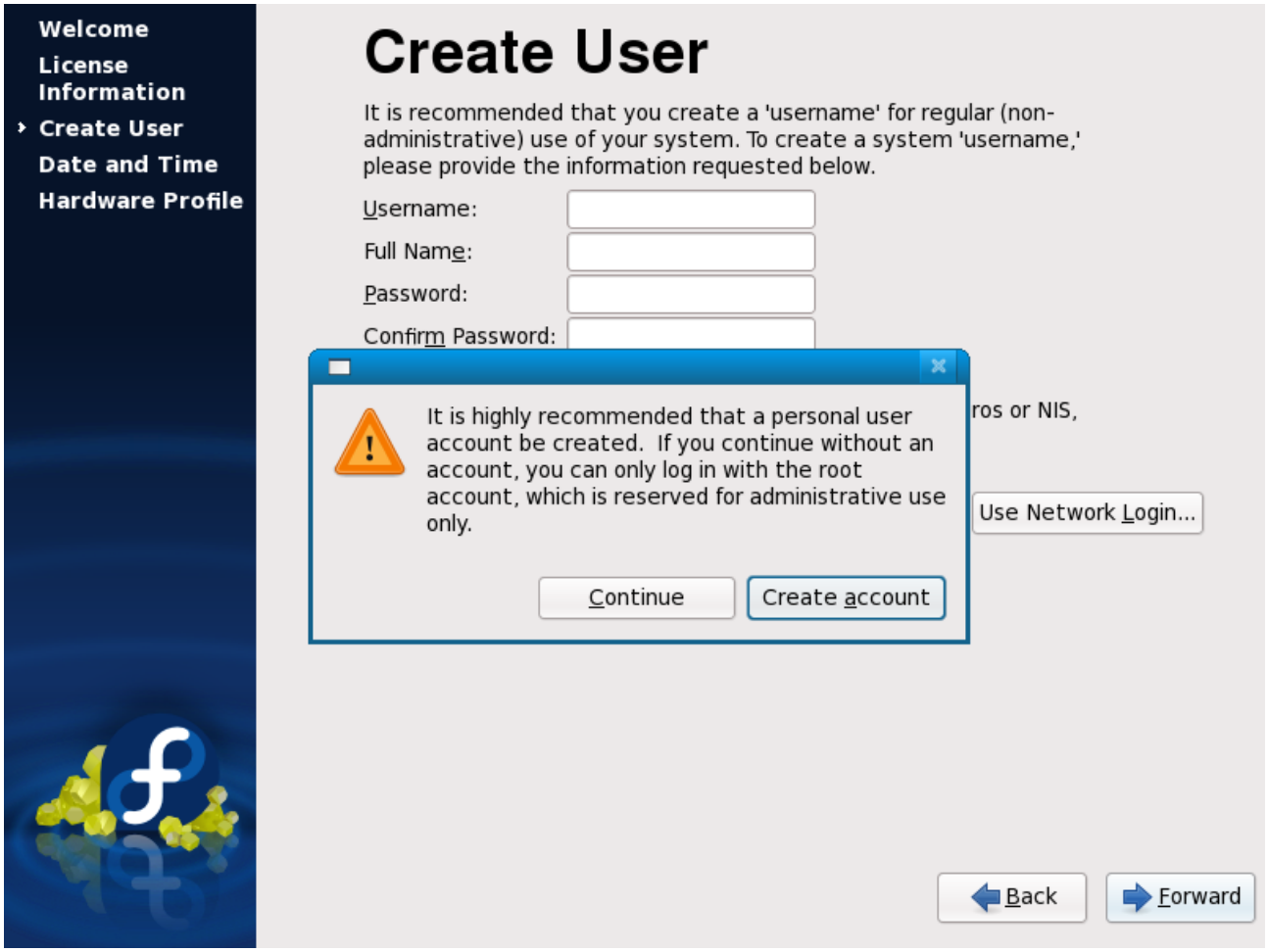
Step16: skip this license page and go “forward”



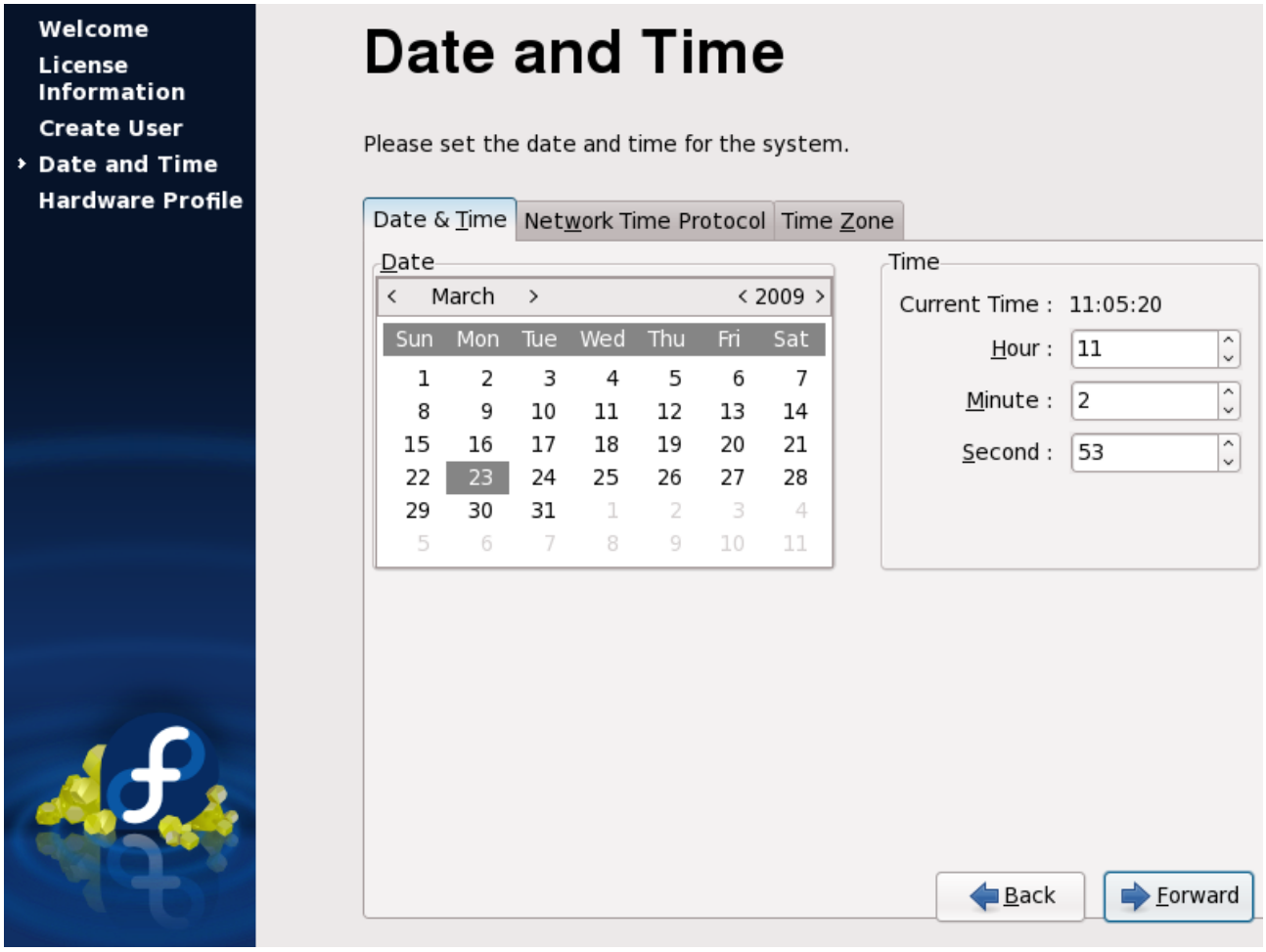
Step17: create new users. We ignored user creation and went to the next step.



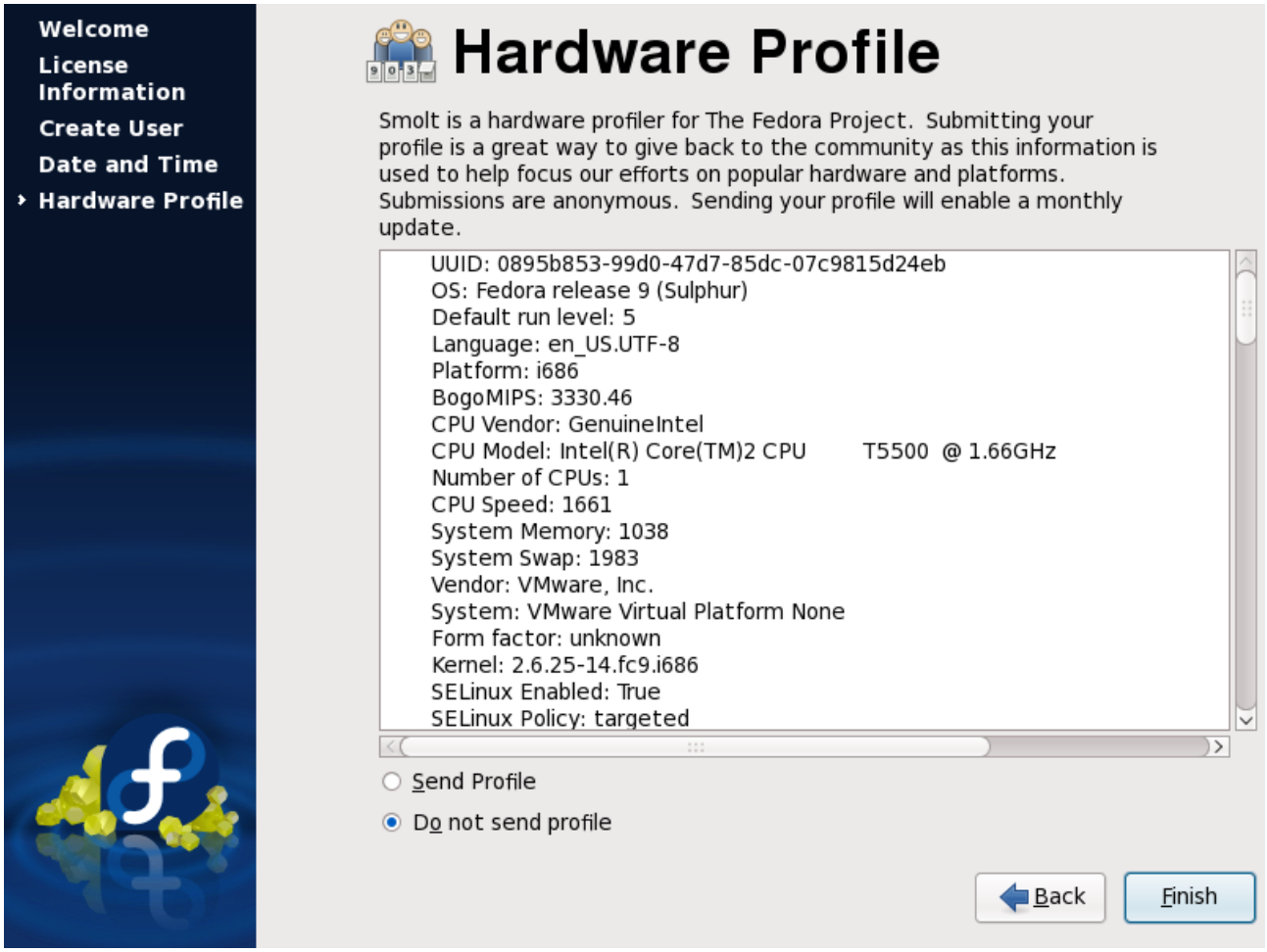
Press “continue” to go on.



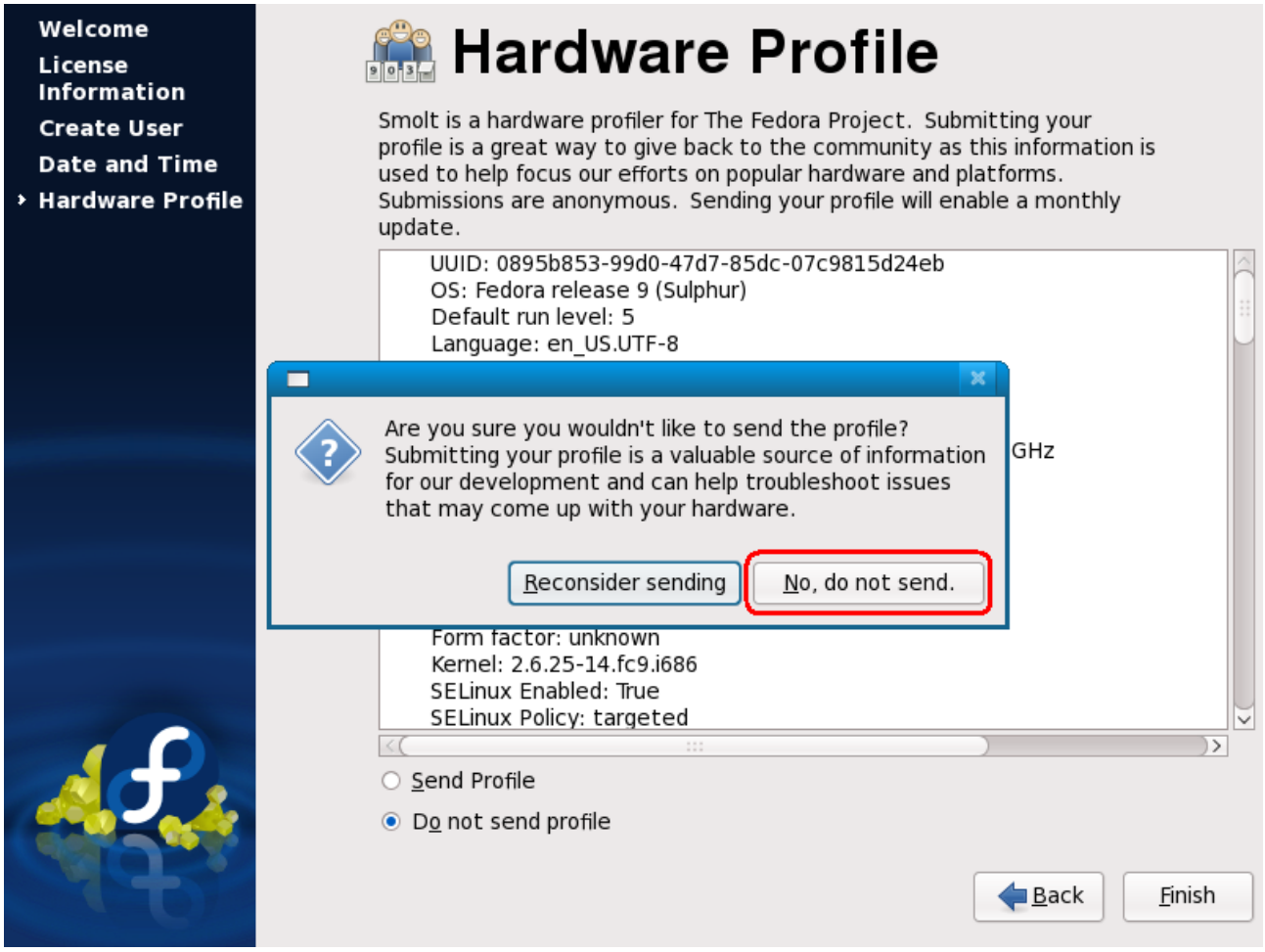
Step18: setup date and time. We ignored this and went to the next step.



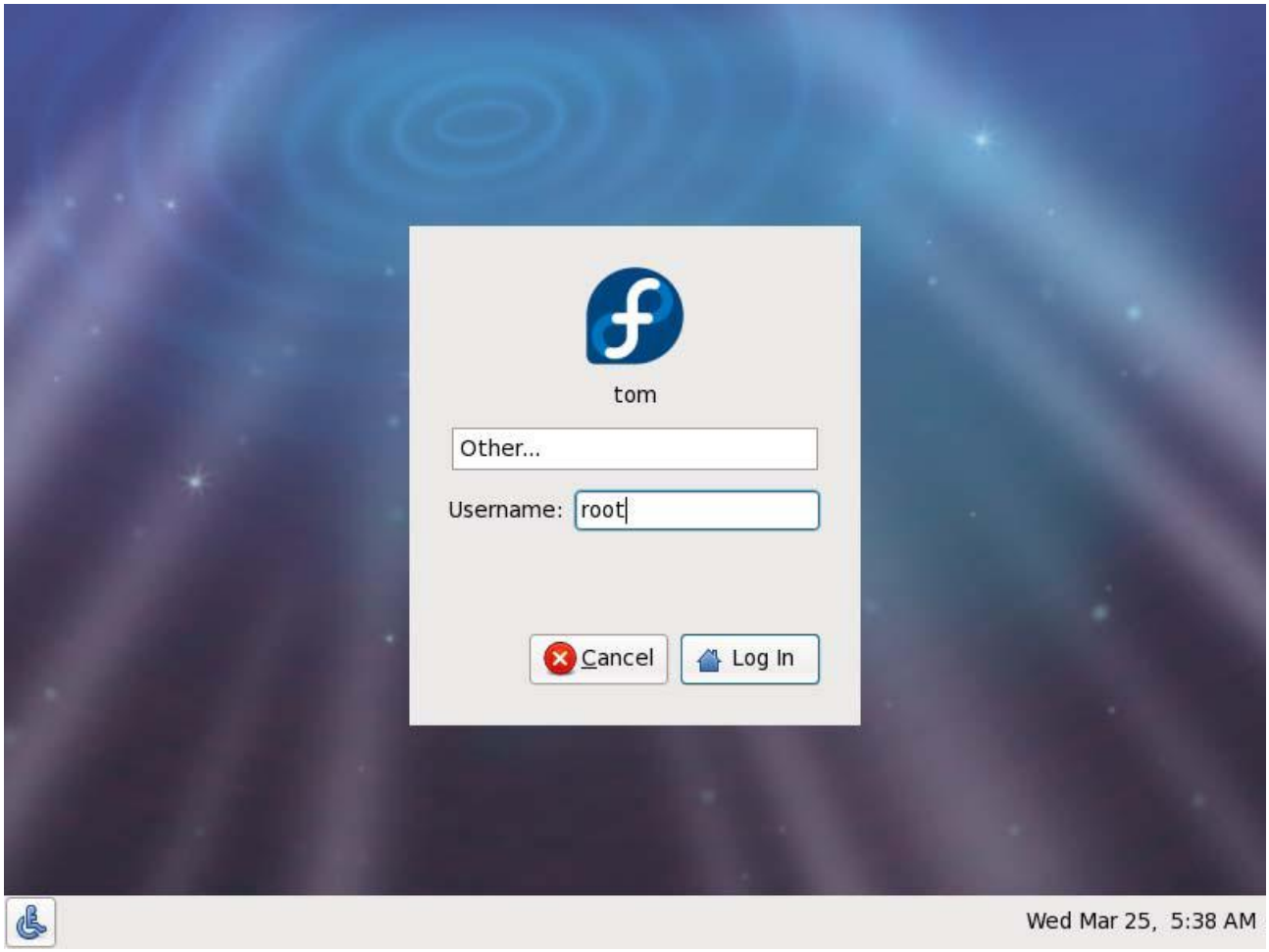
Step19: confirm hardware information. We just clicked on “Finish”.



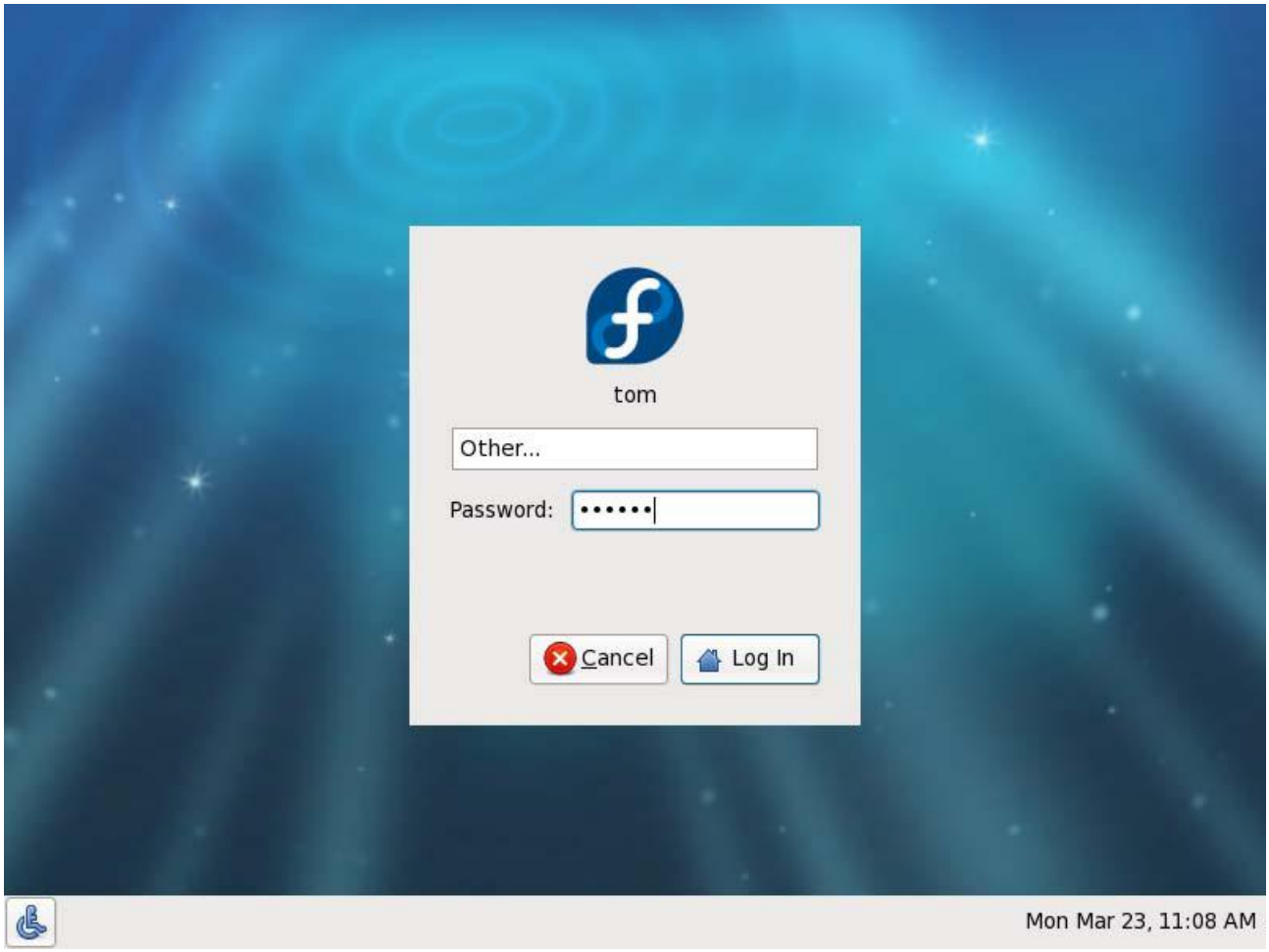
On the popup window shown below, just click on the red marked button.



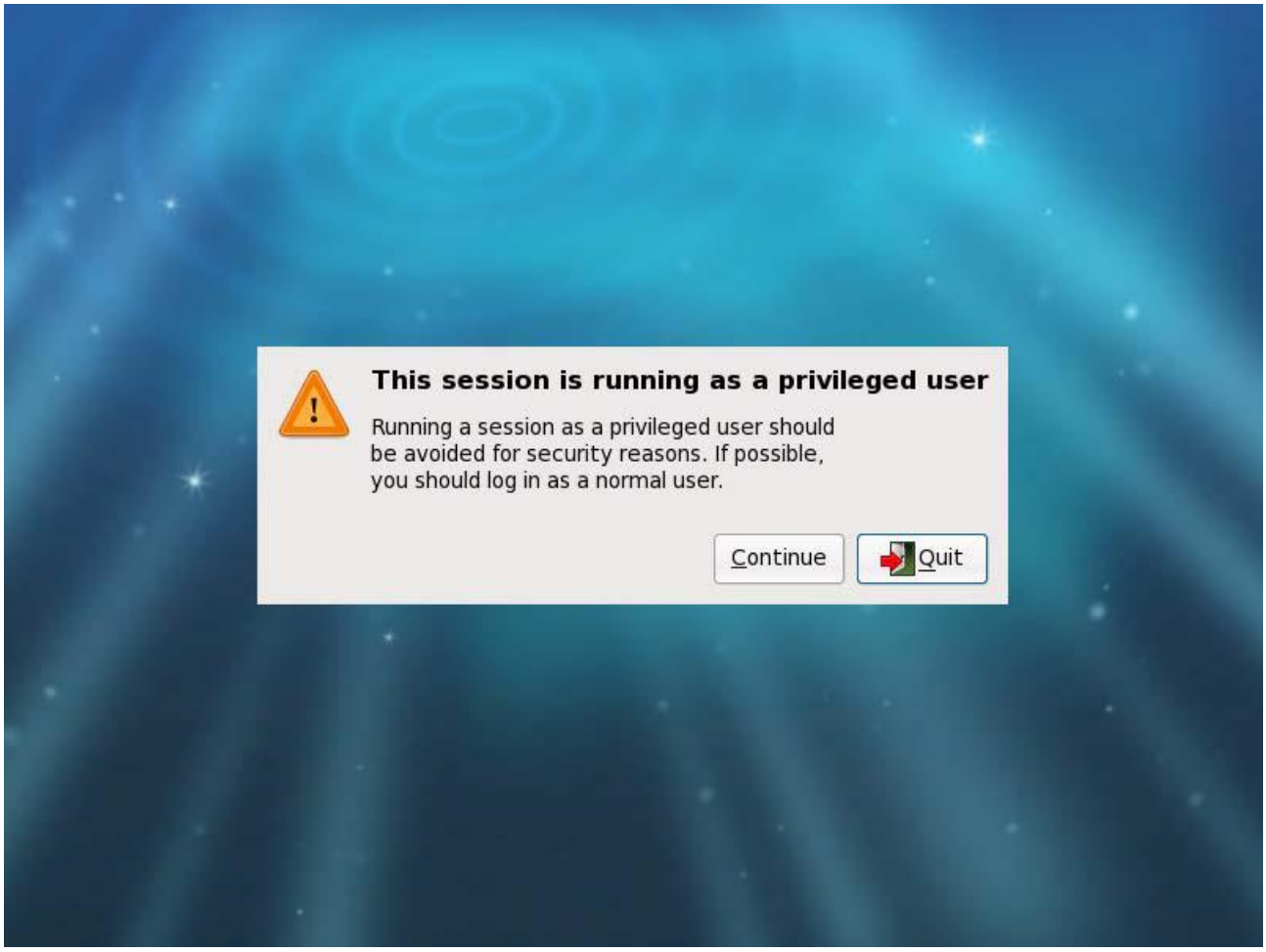
Step 20: on the login page, login as “root”



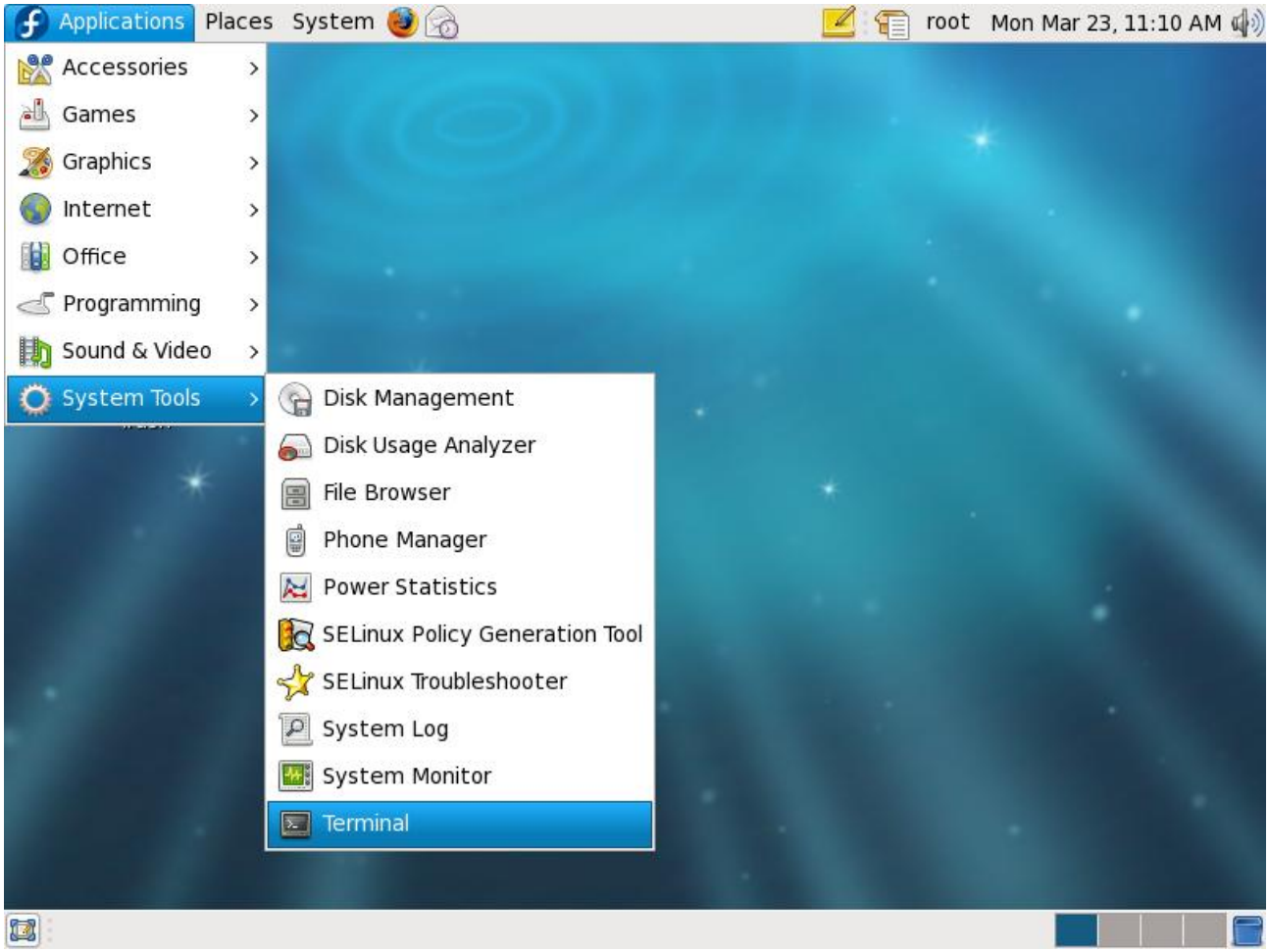
Input the password we just created for “root”



When login as “root”, the following popup window will show up, just click on “Continue”



Below is the interface the user will see after a successful login.

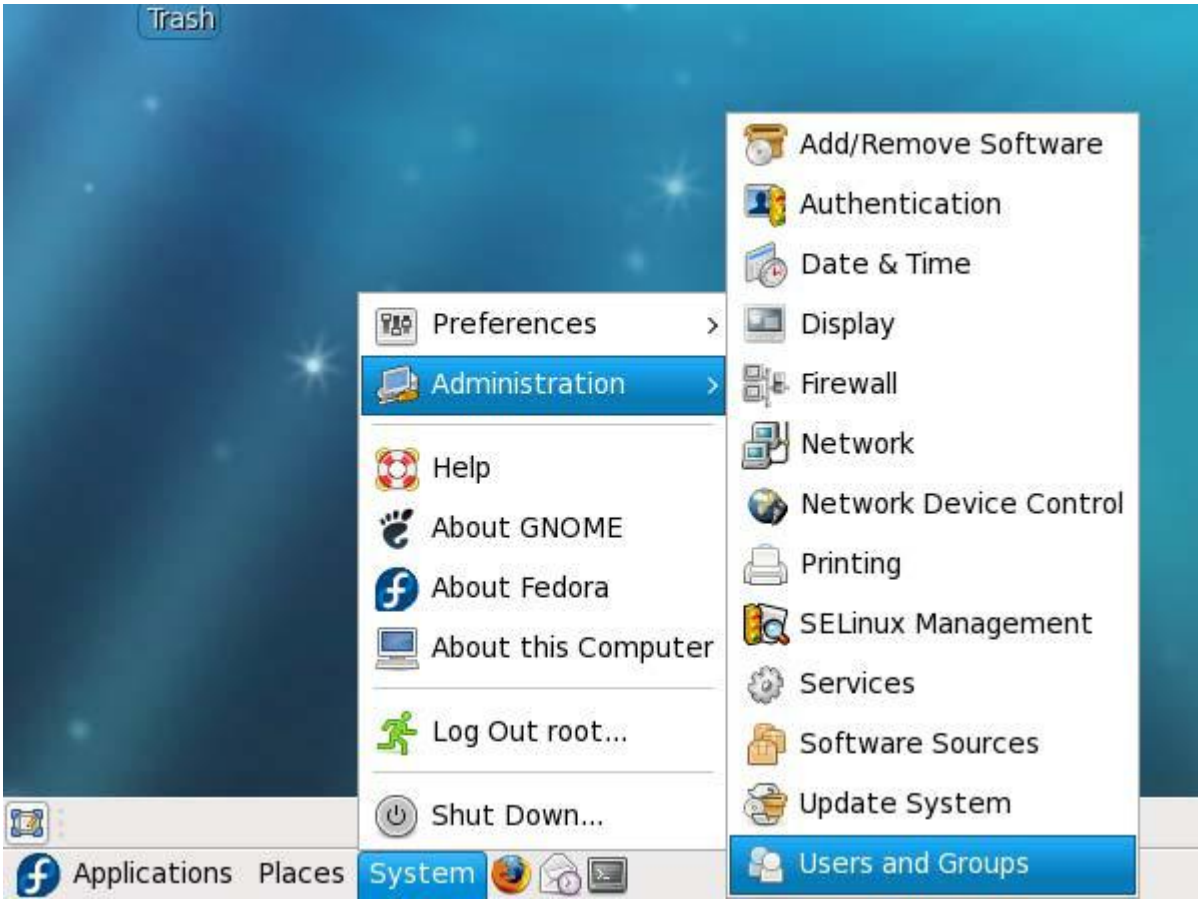


3.2 Basic Settings and Services

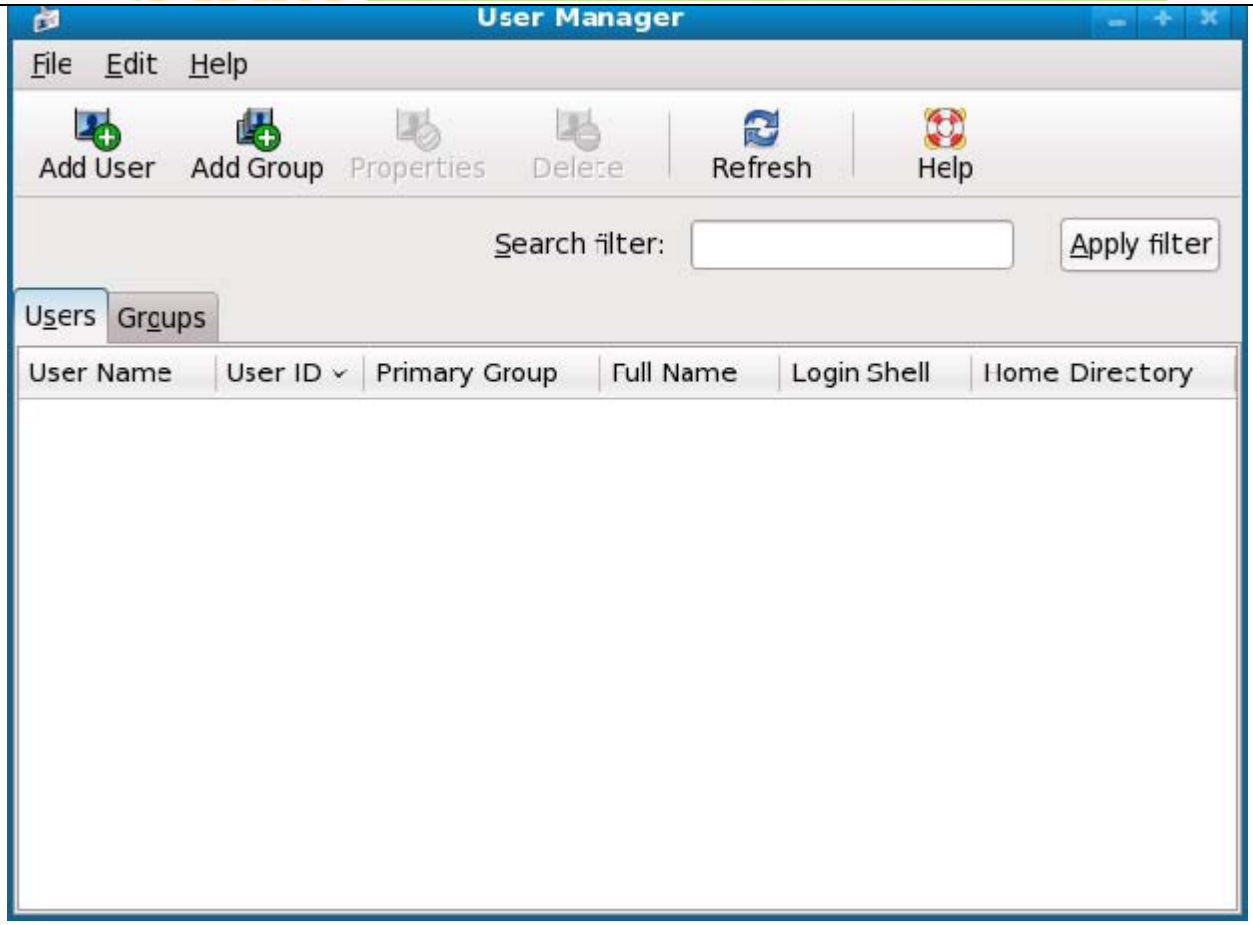
3.2.1 Add User Account

To create a new user (not root) account, here are the steps:

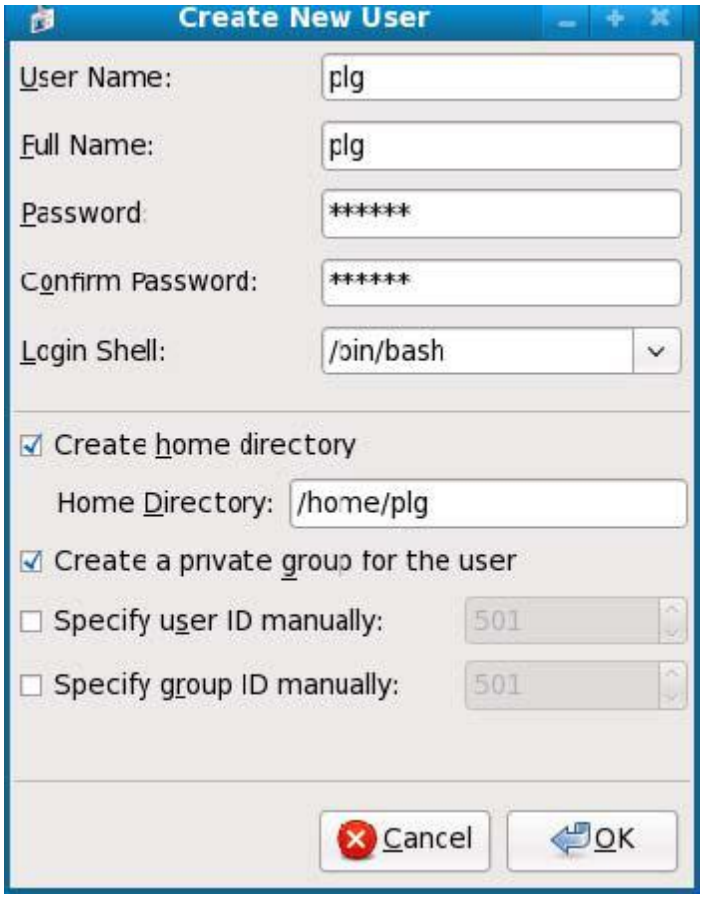
Step 1: go to “Users and Groups”



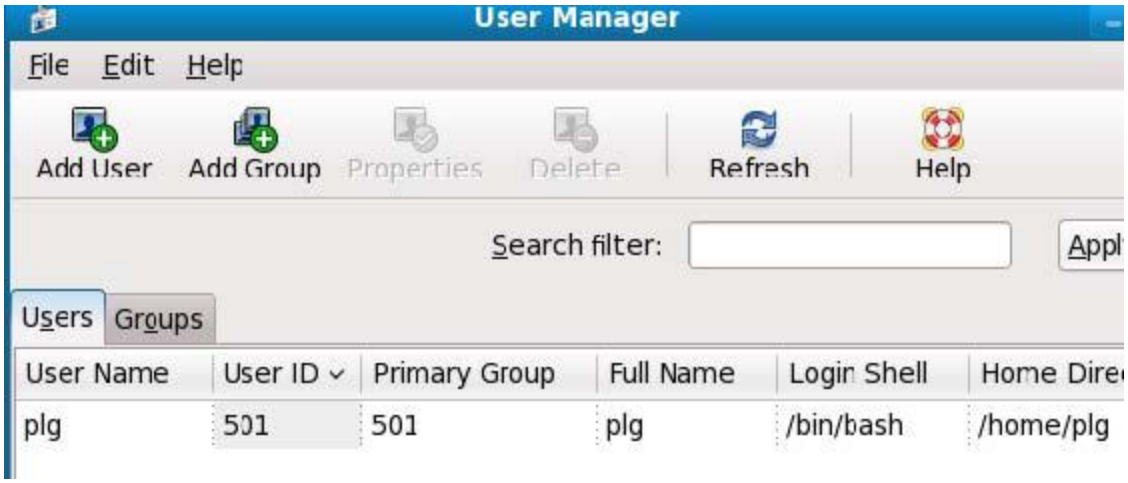
Step 2: open the “Users Manager” window

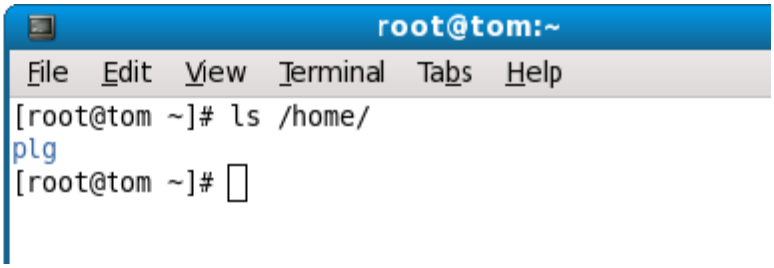


Step 3: click on the “Add User” button, type the user name and password



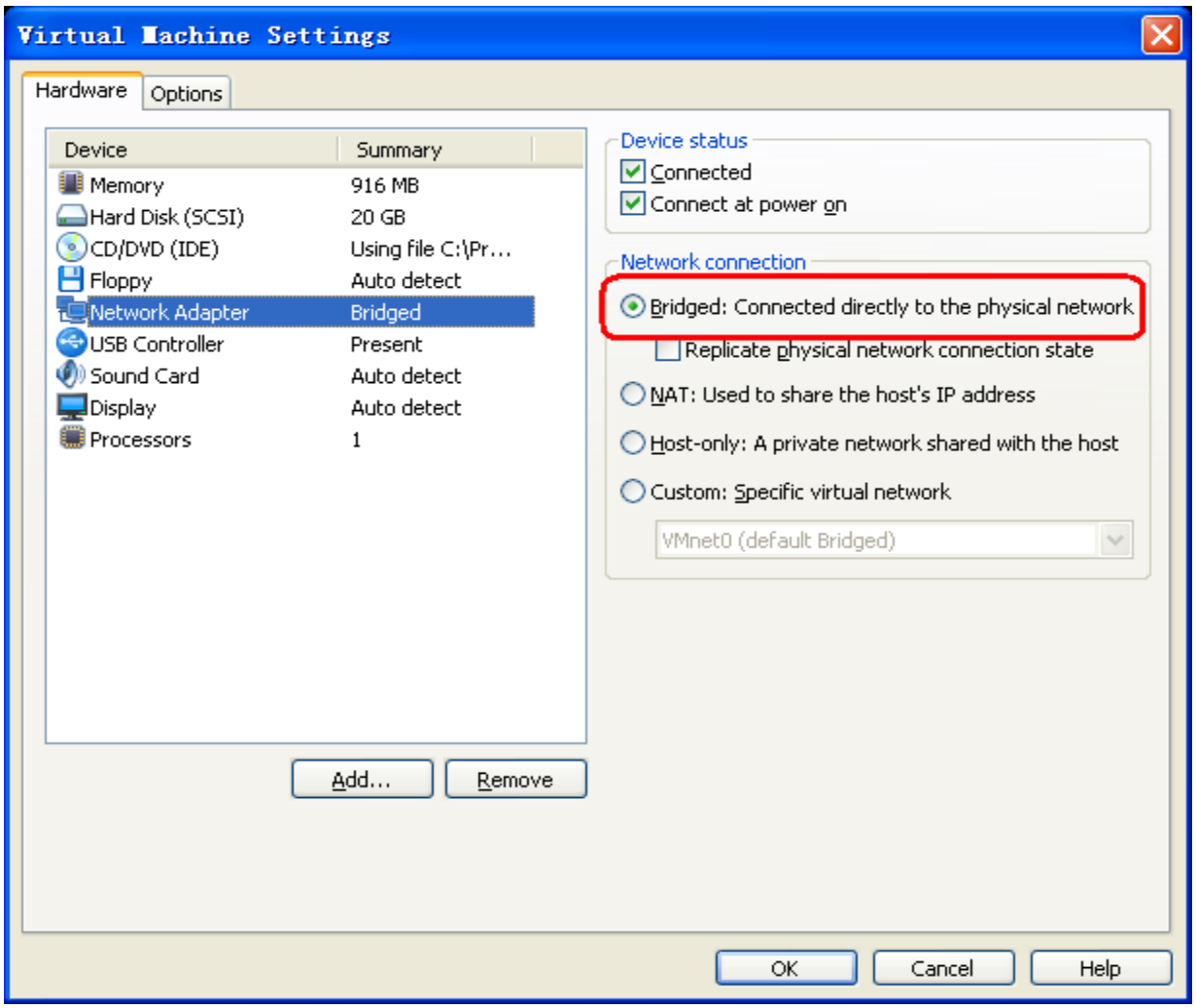
Click on “OK”, you will see that a new “plg” user has been created, and a “plg” directory has been created in the “/home” directory too.





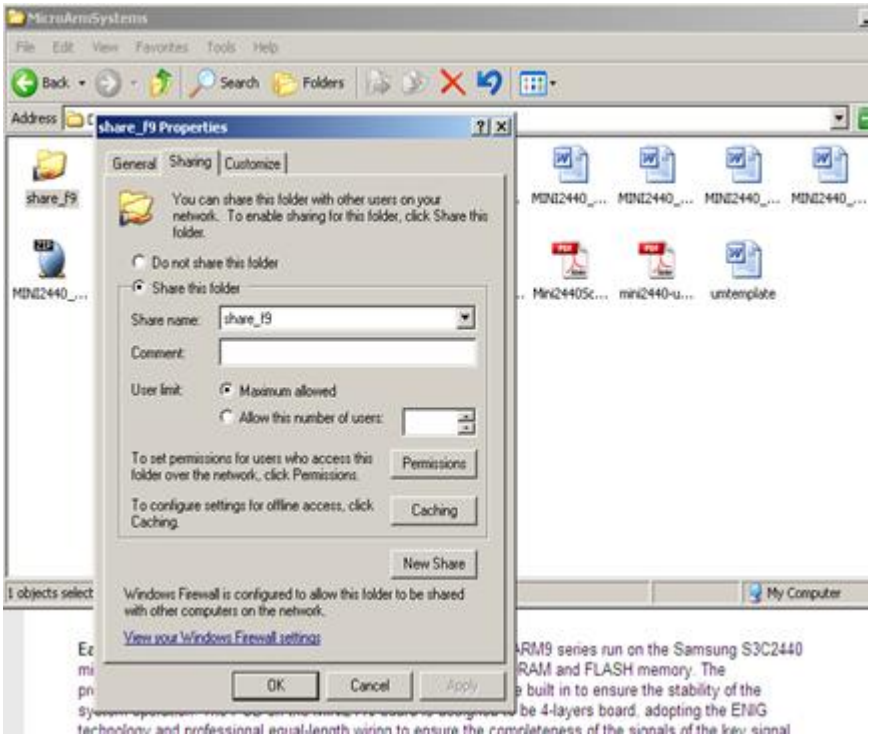
3.2.2 Access Windows Files

You can easily access shared files in Windows from either a virtual machine or a real Fedora9 system as long as they can communicate. To connect to a Windows from a virtual machine, the easiest way is to set “Guest” to “Bridge” in the network configuration.

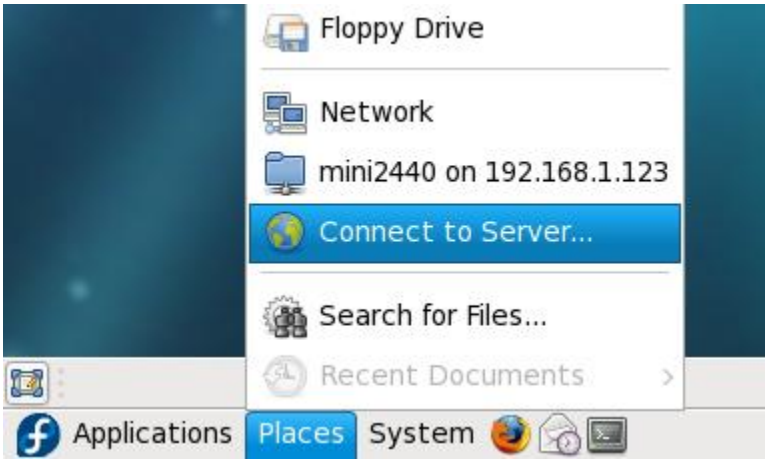


To access shared files in Windows, please following the steps below:

Step 1: set a shared directory in Windows. Here we set a “share_f9”



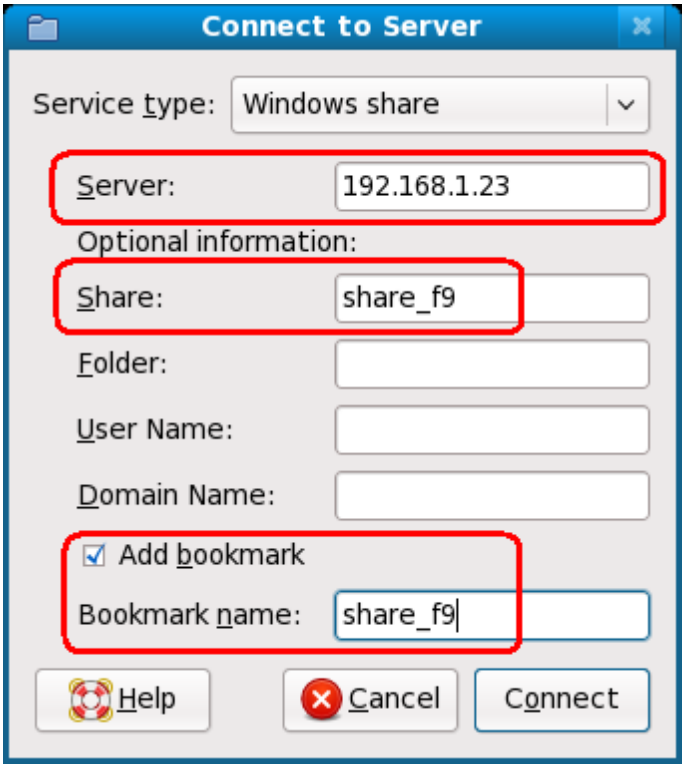
Step 2: set Fedora9



Open the window below:

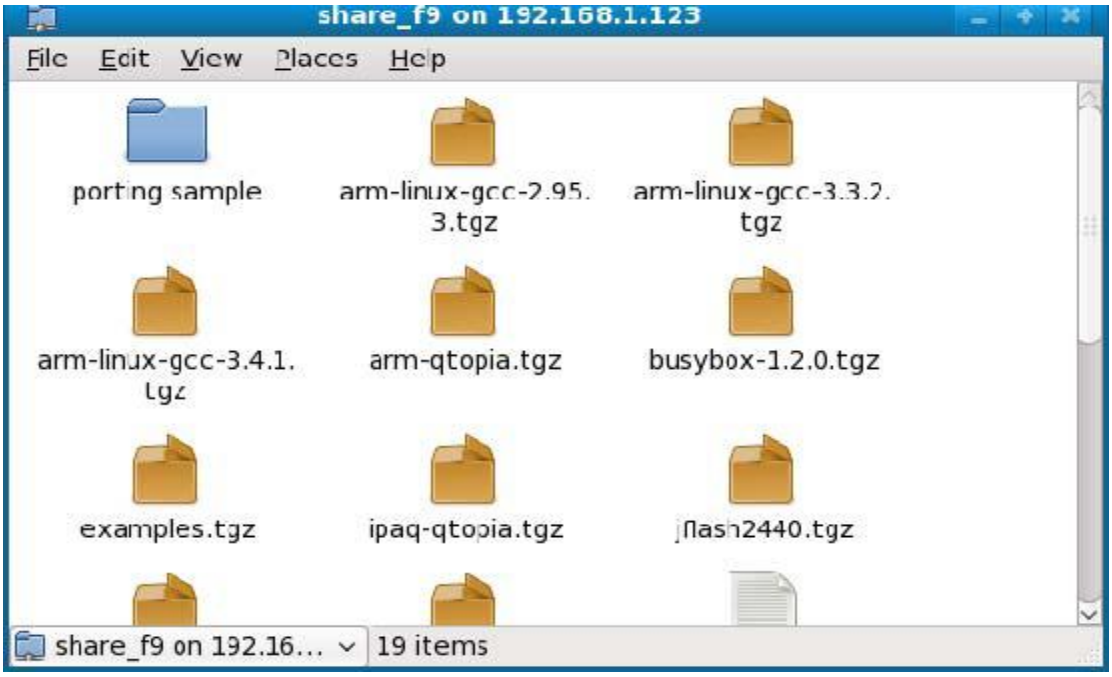
Select “Windows share” in the “service type” field

Input the shared file’s name and its windows machine IP

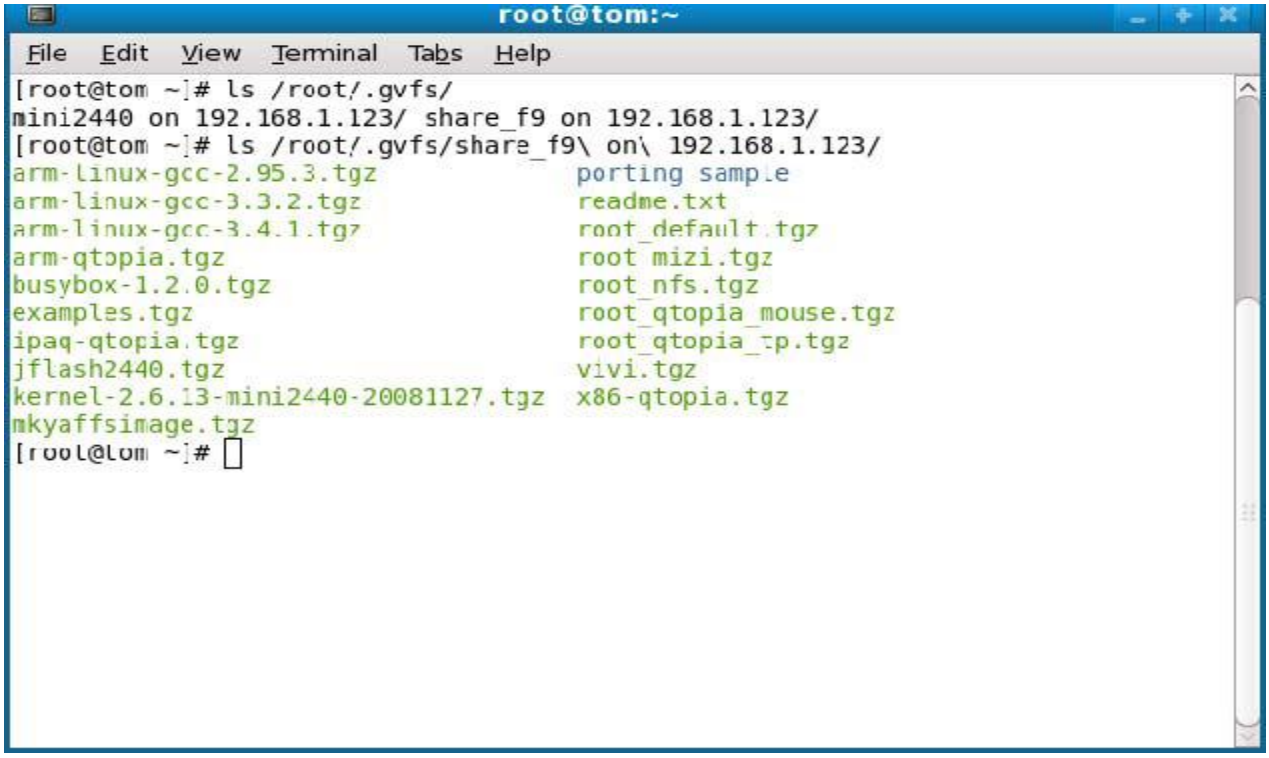


Click on “connect”, the following window will show up:

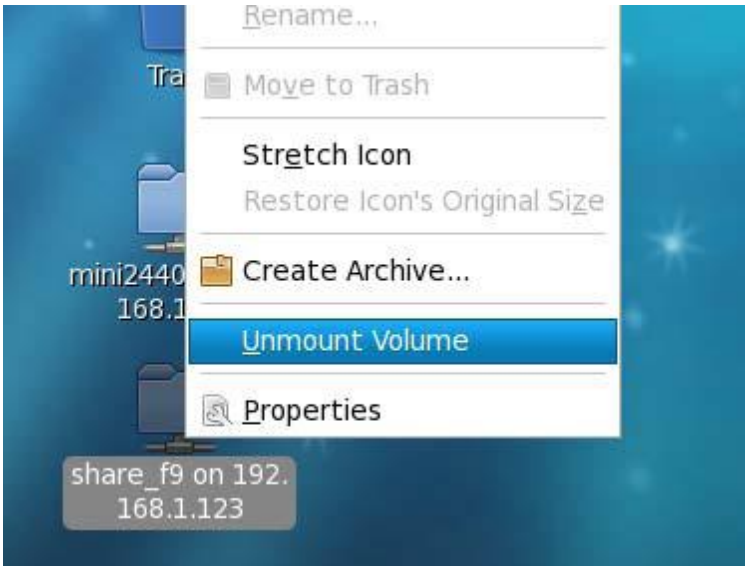
Go ahead and “connect” again, you will see the shared files you just set in your windows system.



If you want to access this directory from the command line utility, you can do it by hitting the TAB key.



To disconnect the shared directory, right click on the shared directory and following the operations in the screenshot below:



3.3 Setup Cross Compile Environment

To compile kernels, Qtopia, bootloader and other programs in Linux you need a cross compile environment. We used arm-linux-gcc-4.4.3. The following steps will introduce how to build a compile environment.

Step 1: copy the compressed file "arm-linux-gcc-4.4.3.tgz" in the shipped CD into a system's directory, e.g "tmp\", enter this directory and execute the following commands:

#cd \tmp

#tar xvzf arm-linux-gcc-4.4.3.tgz –C /

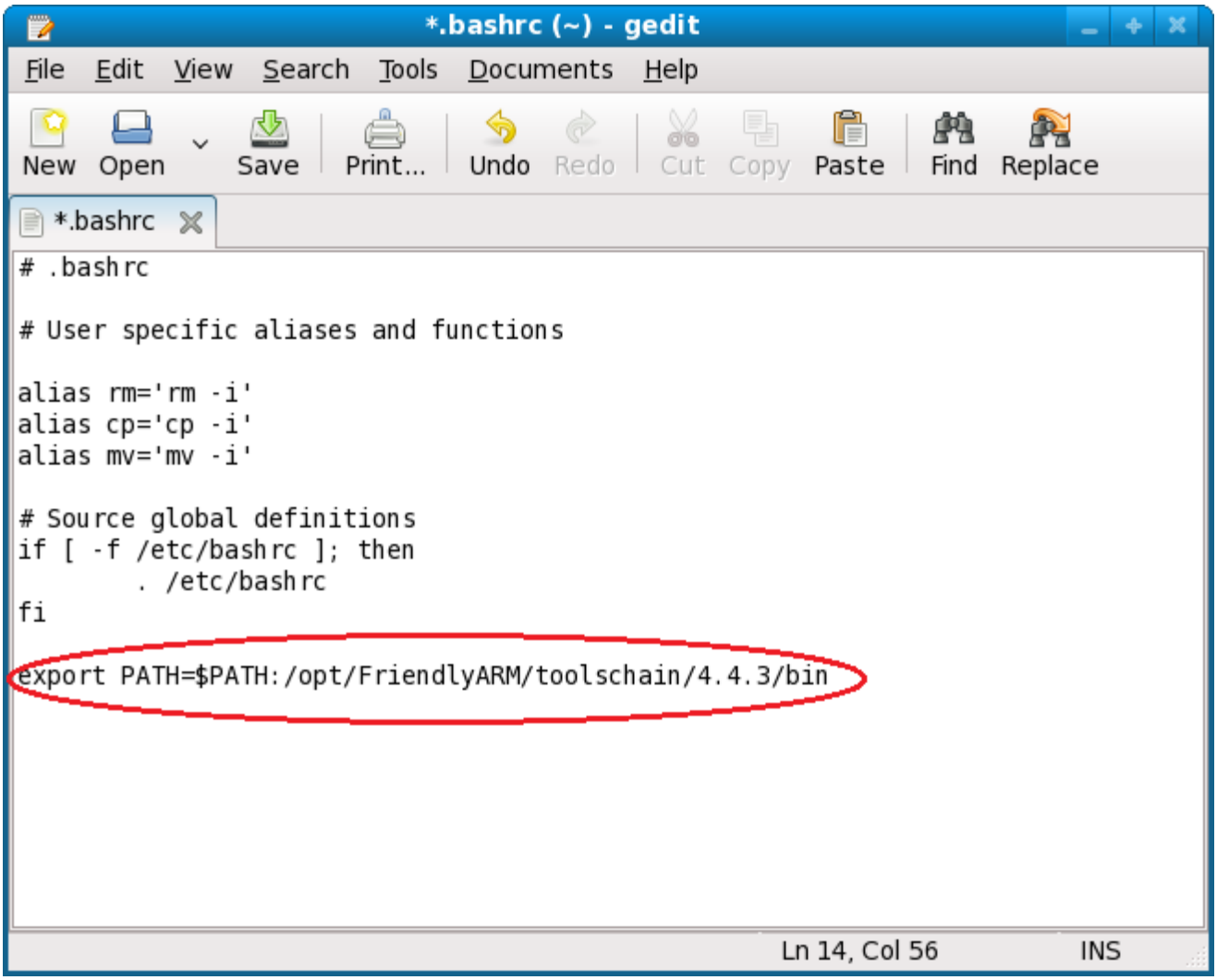
Note: there is a space after "C" and "C" is a capital letter.

These commands will install "arm-linux-gcc" in the "**/opt/FriendlyARM/toolschain/4.4.3**"

Step 2: run the command below to add the compiler's path to system variables:

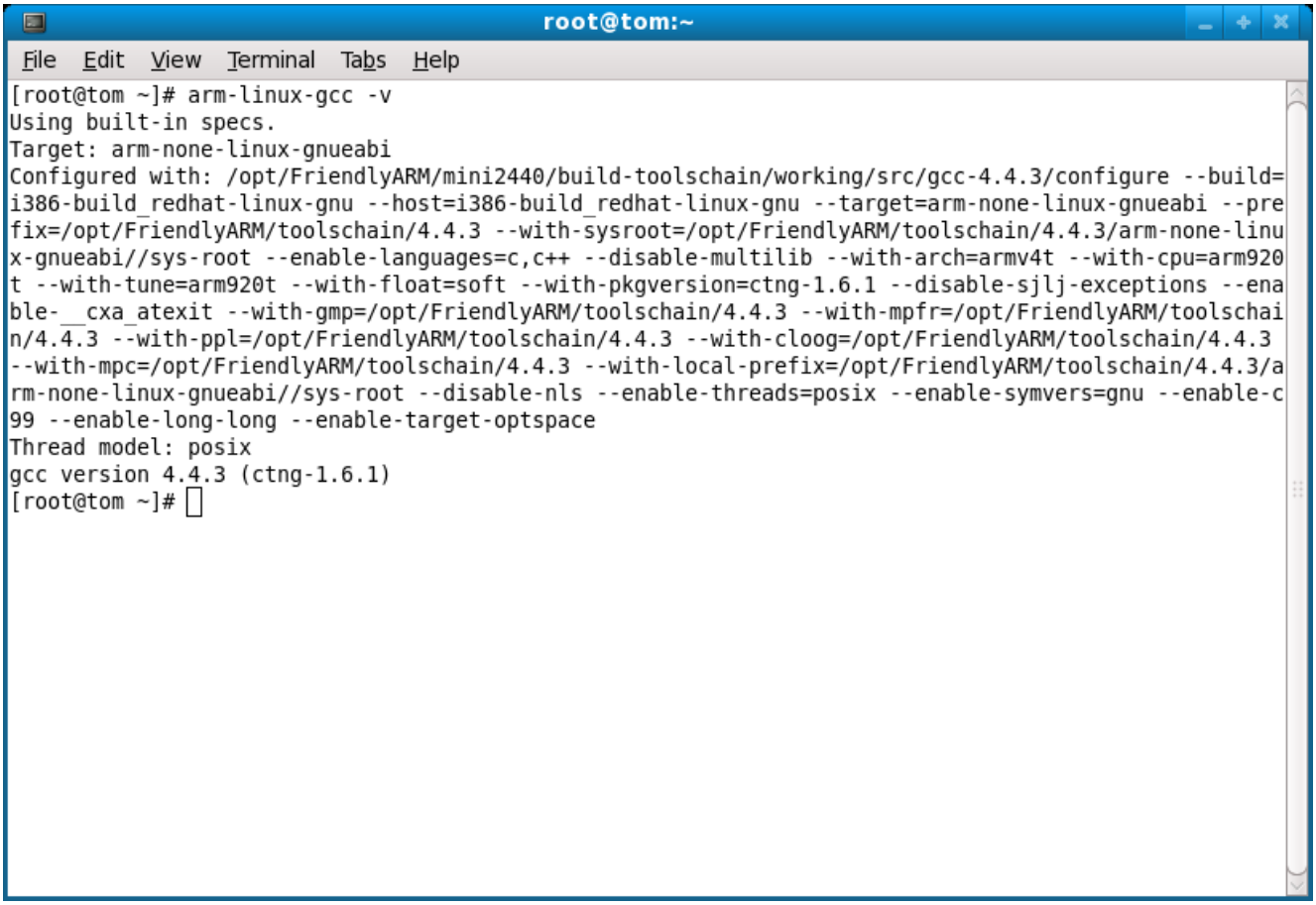
#gedit /root/.bashrc

This is to edit the "/root/.bashrc" file. Update the last line with "**export PATH=$PATH:/opt/FriendlyARM/toolschain/4.4.3/bin**" in the opened file, save and exit the file.



Logout and login the system again (no need to reboot the system, just go to "start"-> "logout") the above settings will take into effect. Type "arm-linux-gcc –v", if the messages depicted in the screen shot below appear it indicates the compile environment

has been set up successfully.



3.4 Uncompress Source Code and Install Application Utilities

This section will introduce how to uncompress all the source code that users may need and install some application utilities including:

- Linux kernel source code
- Qtopia-2.2.0 source code (for x86 and arm)
- QtE-4.8.5 (for ARM)
- Busybox-1.13 source code
- Sample programs code (developed by FriendlyArm)

- Target file system directory
- File system image maker (for YAFFS2 and UBIFS)
- Linux logo maker: logo_maker

Note: all source code and utilities should be uncompressed and compiled with arm-linux-gcc-4.4.3

3.4.1 Uncompress Source Code

Firstly, create a working directory: /opt/FriendlyARM/mini2451

After execute command "mkdir –p /opt/FriendlyARM/mini2451", all the source code in the following steps will be uncompressed in this work directory

(1) Get Linux source code ready

In Fedora9, create a temporary director "/tmp/" by running the following command

#mkdir /tmp

Copy all the files in the linux directory in the shipped CD to "/tmp"

(2) Uncompress the Linux kernel source code

In the work directory /opt/FriendlyARM/mini2451, run the commands below:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/linux-3.6-20130826.tar.gz

A linux-3.6 directory will be created it includes a complete copy of linux kernel source code.

Note: 20130826 is the date when FriendlyARM released the new version, the file name

in the shipped CD may be different.

(3) Uncompress and install Qtopia source code

In the work directory /opt/FriendlyARM/mini2451, run the commands below:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/x86-qtopia-20100420.tar.gz

#tar xvzf /tmp/linux/arm-qtopia-20100108.tar.gz

An x86-qtopia directory and an arm-qtopia directory will be created, and their source code will be uncompressed into these two directories.

Note: in this release, supports for mouse and tp are all included in one package.

(4) Uncompress and install QtE-4.8.5 source code

In the work directory /opt/FriendlyARM/mini2451, run the commands below:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/arm-qte-4.8.5-20130922.tar.gz

An arm-qte-4.8.5 will be created, and their source code will be uncompressed into it.

(5) Uncompress and install busybox source code

The Busybox is a compact Linux tool kit. Here we used busybox-1.20.2. Users can download its latest version from http://www.busybox.net

In the work directory /opt/FriendlyARM/mini2451, run the commands below:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/busybox-1.20.2-Mini2451.tgz

A busybox-1.20.2 directory will be created, and its source code will extracted into

this directory.

Note: for the sake of users, we have made a default configuration file: fa.config.

(6) Uncompress and install Linux sample programs

In the work directory /opt/FriendlyARM/mini2451, run the commands below:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/examples-20100108.tgz

An examples directory will be created, all the source code will be extracted into this directory.

Note: all these sample programs are developed by FriendlyARM.

3.4.2 Create Target File System

Please execute the following commands:

#cd /opt/FriendlyARM/mini2451

#tar xvzf /tmp/linux/ rootfs_qtopia_qt4-20130918.tgz

A rootfs_qtopia_qt4 will be created.

This package includes qtopia-2.2.0, busybox and some command line utilities. It has the following excellent features:

- auto detection of NFS reboot or local reboot
- auto detection of touch screen and launching the calibration utility if necessary. If no touch screen is connected system will enable the mouse.
- auto detection of command or high speed SD cards (up to maximum memory of 32G)

and flash drives

- auto detection of USB mouse or touch screen

- include Qtopia-2.2.0 and QtE-4.8.5

3.4.3 Install Utilities

(1) Install mkyaffs2image

To burn a target file system to the board you need to make the file system an image first. We have a tool that can be used to make file images: mkyaffs2image. Below are the steps to install:

#cd /

#tar xvzf /tmp/linux/mktools-20130822-2451.tar.gz

This will create those tools in the "/usr/sbin" directory.

Note: "C" is capitalized and means "change". If your system has been installed a Mini2440's mkyaffs2image it will be overwritten. But you don't need to worry about it since they are identical

(2) Install LogoMaker

LogoMaker is developed by FriendlyARM for making linux logos. There are many resources describing how to convert image files such as bmp, jpg, png and so on to linux logos using command line tools. We created this graphic version which is based on Fedora9.

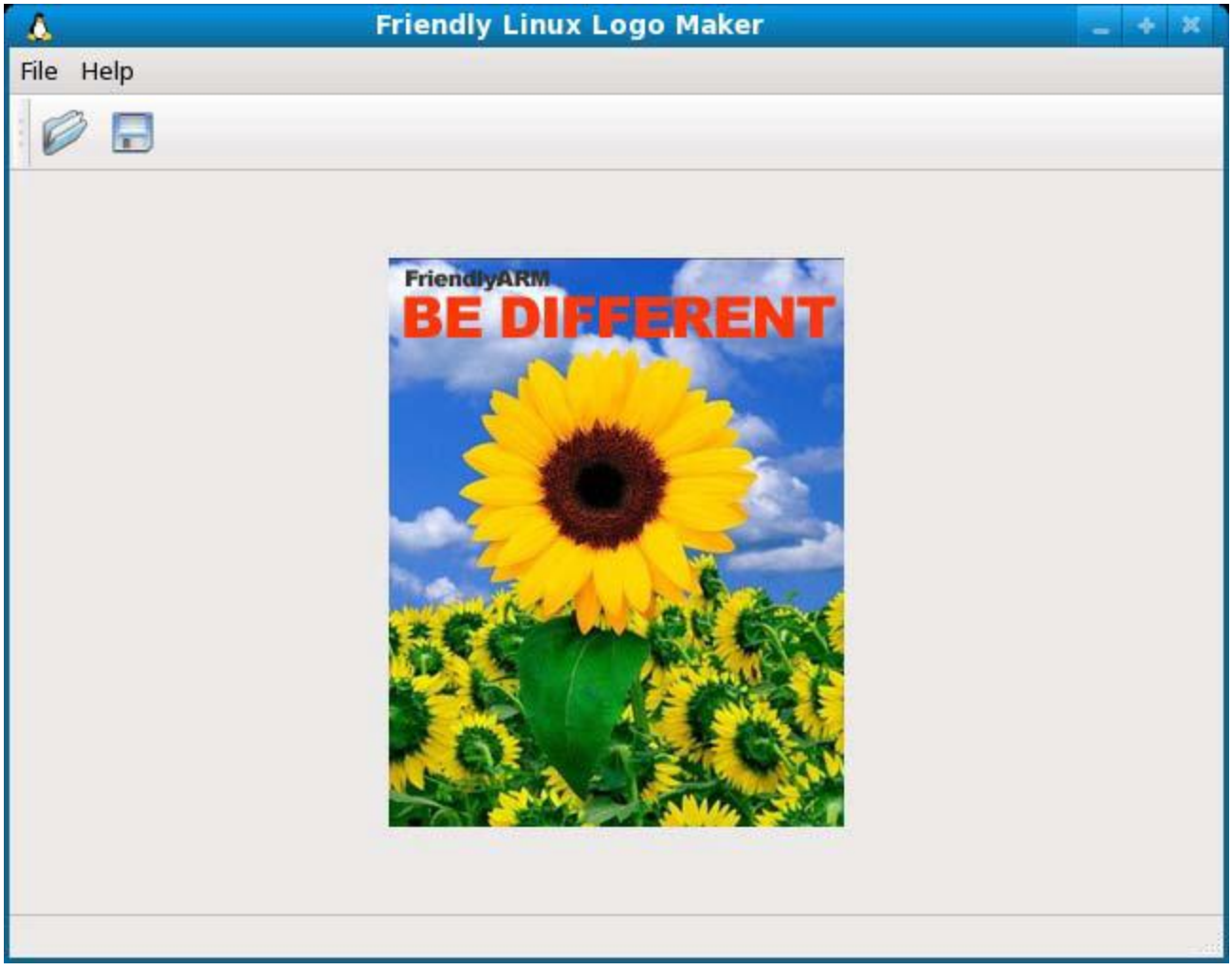
Execute the command below:

#cd /

#tar xvzf /tmp/linux/logomaker.tgz –C /

Note: “C” is capitalized and means “change”.

After executing the above commands, LogoMaker will be installed in the /usr/sbin directory. It only has one file. After installing it, type “logomake” in a command line window, you will see the following screenshot



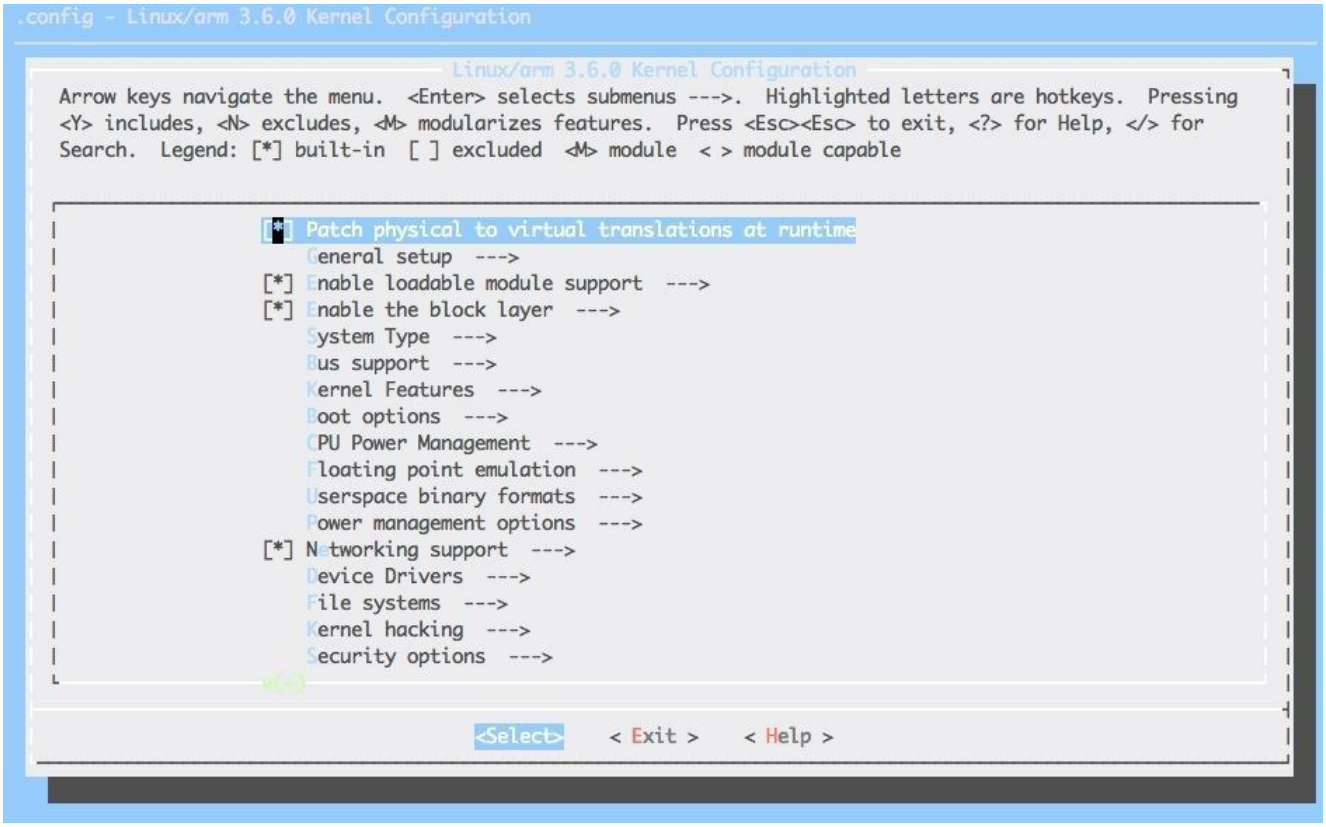
4 Make Linux Kernel and File System

4.1 Compile Linux Kernel

Type the following command to compile:

#cp mini2451_linux_config .config : there is a space after "config" and prior to "."

#make menuconfig ; begins to compile



Actually you don't need to do anything just select <Exit> a configuration file will be generated.Then please command:

#make

After the compilation is done, an image file **zImage** will be generated under "**arch/arm/boot**".

4.2 Make File System Image

Please make sure you have installed the mkyaffs2image tool.

Enter "/opt/FriendlyARM/mini2451" and execute the following command:

#mkyaffs2image rootfs_qtopia_qt4 rootfs_qtopia_qt4.img

This will compress the whole "rootfs_qtopia_qt4" into a yaffs2 rootfs_qtopia_qt4.img file. It is the same as the one in "/images/Linux/" in the shipped CD. Download it to your board's NAND Flash.

5 Development with Linux

This section lists some sample Linux programs for users' reference. All the following programs are compiled with arm-linux-gcc-4.4.3. We don't guarantee they can be compiled and run with other corss compilers. To check your compiler, please type "arm-linux-gcc –v"

5.1 Hello World

5.1.1 Souce Code

The source code of "Hello,World" is under "**/opt/FriendlyARM/mini2451 /examples/hello**". Its contents are as follows:

```
#include <stdio.h>
int main(void) {
printf("hello, FriendlyARM!\n");
}
```

5.1.2 Compile

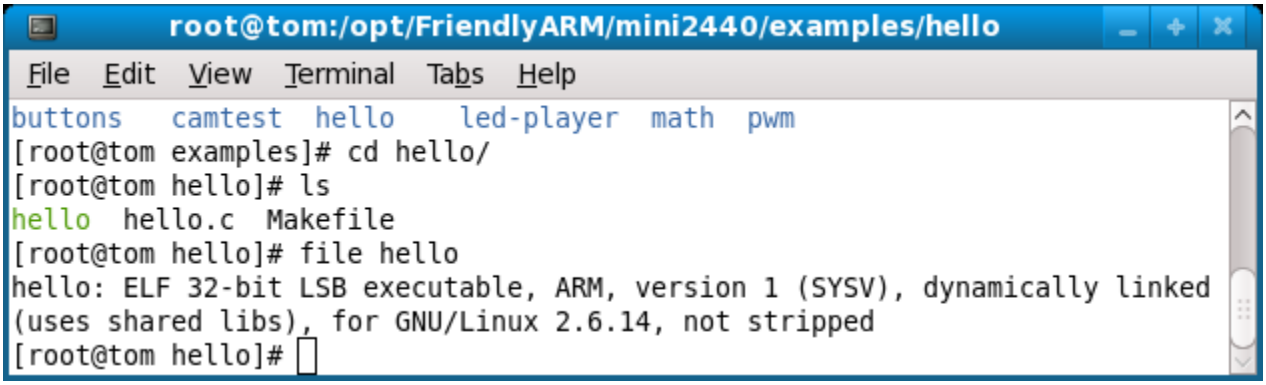
Enter the directory where the source code is located and execute "make":

#cd /opt/FriendlyARM/mini2451/ examples/hello

#arm-linux-gcc –o hello main.c Or type the command below

#make

A "hello" executable will be generated and you can check whether it is for ARM by commanding "file":



5.1.3 Download and Run

You can download your executable to the board in any of the following ways:

- Copy to a media (such as flash drives)
- FTP file transfer (recommended)
- File transfer via serial port
- Run via NFS

(1) Copy to Flash Drive

Note: copy your executable to a flash drive, mount it to your board and copy the file to "/bin"

1. Copy to Flash Drive

Connect your flash drive to your PC and execute the following commands

#mount /dev/sda1 /mnt ; mount your drive

#cp hello /mnt ; copy your file to the drive

#umount /mnt ; unmount your drive

2. Copy to Board

Insert your drive to your board's USB host it will be automatically mounted under "/udisk". Please execute the following command

#cd /udisk

#./hello ； execute "hello"

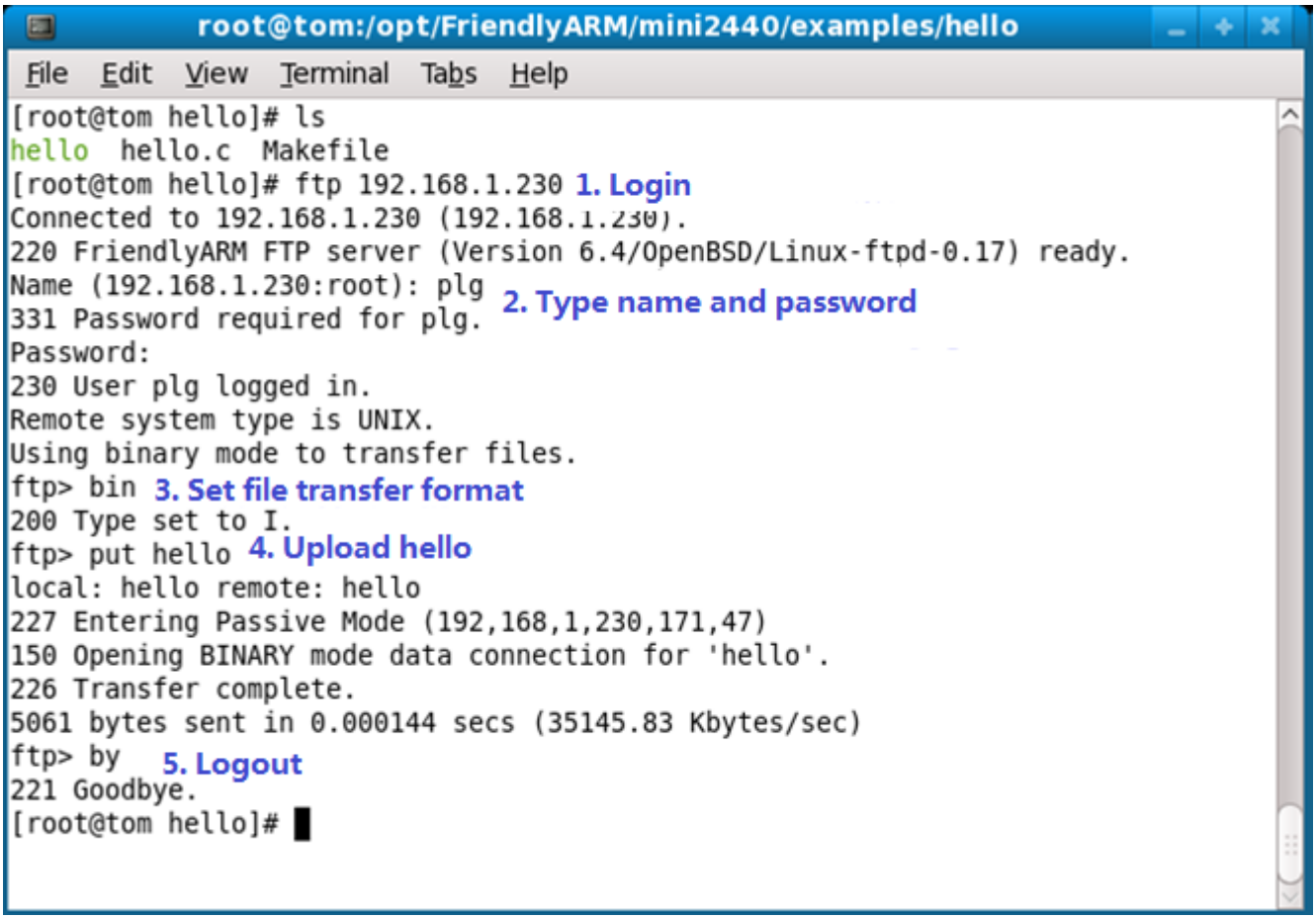
Note: if you take out your drive directly you need to go back to the root directory and execute "umount /udisk" for the next mount

```
usb 1-1: Product: DataTraveler 2.0
usb 1-1: Manufacturer: Kingston
usb 1-1: SerialNumber: 001AA0A0BF1AC8C1155A0318
usb 1-1: configuration #1 chosen from 1 choice
scsi1 : SCSI emulation for USB Mass Storage devices
scsi 1:0:0:0: Direct-Access     Kingston DataTraveler 2.0 1.00 PQ: 0 ANSI: 2
sd 1:0:0:0: [sda] 7823296 512-byte hardware sectors: (4.00 GB/3.72 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] 7823296 512-byte hardware sectors: (4.00 GB/3.72 GiB)
sd 1:0:0:0: [sda] Write Protect is off
sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 1:0:0:0: [sda] Attached SCSI removable disk
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will b
e case sensitive!

[root@FriendlyARM /]# cd /udisk/
[root@FriendlyARM /udisk]# ls
hello   images  linux   mp3     photo   video
[root@FriendlyARM /udisk]# ./hello
hello, FriendlyARM!
[root@FriendlyARM /udisk]# _
```

(2) FTP File Transfer

Note: login your board via FTP, transfer your executable to it and change its file property to executable

First, execute your commands in PC



Go to your board and execute the following commands:

5.2 Sample Application Code

5.2.1 LED Test Program

The source code of the LED test program is under

"**/opt/FriendlyARM/mini2451/linux-3.6/drivers/char**". Its contents are as follows:

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char
Driver	Mini2451_leds.c
Device Type	misc
Device Name	/dev/leds
Test Program Source Code Location	/opt/FriendlyARM/mini2451/ examples/leds
Test Program Name	led.c

Executable Name	led
Test Program's Location in Board	
Note: the LED driver has been compiled into the kernel by default and you cannot load it via insmod	

Program:

```
#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;
/* Check parameters */
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) {
fprintf(stderr, "Usage: leds led_no 0|1\n");
exit(1);
}
/*Open /dev/leds file*/
fd = open("/dev/leds0", 0);
if (fd < 0) {
fd = open("/dev/leds", 0);
}
if (fd < 0) {
perror("open device leds");
exit(1);
}
/*Manipulate led via ioctl and input parameters */
ioctl(fd, on, led_no);
/*Close device*/
close(fd);
return 0;
}
```

You can compile the program, download it and run

5.2.2 User Button Test Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char

Driver	Mini2451_buttons.c
Device Type	misc
Device Name	/dev/buttons
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/buttons
Test Program Name	button_test.c
Executable Name	buttons
Test Program's Location in Board	

Note: the button driver has been compiled into the kernel by default and you cannot load it via insmod

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
int main(void)
{
int buttons_fd;
char buttons[6] = {'0', '0', '0', '0', '0', '0'};
buttons_fd = open("/dev/buttons", 0);
if (buttons_fd < 0) {
perror("open device buttons");
exit(1);
}
for (;;) {
char current_buttons[6];
int count_of_changed_key;
int i;
if (read(buttons_fd, current_buttons, sizeof current_buttons) != sizeof current_buttons)
{
perror("read buttons:");
exit(1);
}
for (i = 0, count_of_changed_key = 0; i < sizeof buttons / sizeof buttons[0]; i++) {
if (buttons[i] != current_buttons[i]) {
```

```
buttons[i] = current_buttons[i];
printf("%skey %d is %s", count_of_changed_key? ", ": "", i+1, buttons[i] == '0' ? "up" :
"down");
count_of_changed_key++;
}
}
if (count_of_changed_key) {
printf("\n");
}
}
close(buttons_fd);
return 0;
}
```

You can compile the program, download it and run

5.2.3 PWM Buzzer Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char
Driver	Mini2451_pwm.c
Device Type	misc
Device Name	/dev/pwm
Test Program Source Code Location	/opt/FriendlyARM/mini2451/ examples/pwm
Test Program Name	pwm_test.c
Executable Name	Pwm_test
Test Program's Location in Board	
Note: the pwm driver has been compiled into the kernel by default and you cannot load it via insmod	
Program:	

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#define PWM_IOCTL_SET_FREQ 1
#define PWM_IOCTL_STOP 2
#define ESC_KEY 0x1b
static int getch(void)
{
struct termios oldt,newt;
int ch;
```

```
if (!isatty(STDIN_FILENO)) {
fprintf(stderr, "this problem should be run at a terminal\n");
exit(1);
}
// save terminal setting
if(tcgetattr(STDIN_FILENO, &oldt) < 0) {
perror("save the terminal setting");
exit(1);
}
// set terminal as need
newt = oldt;
newt.c_lflag &= ~( ICANON | ECHO );
if(tcsetattr(STDIN_FILENO,TCSANOW, &newt) < 0) {
perror("set terminal");
exit(1);
}
ch = getchar();
// restore termial setting
if(tcsetattr(STDIN_FILENO,TCSANOW,&oldt) < 0) {
perror("restore the termial setting");
exit(1);
}
return ch;
}
static int fd = -1;
static void close_buzzer(void);
static void open_buzzer(void)
{
fd = open("/dev/pwm", 0);
if (fd < 0) {
perror("open pwm_buzzer device");
exit(1);
}
// any function exit call will stop the buzzer
atexit(close_buzzer);
}
static void close_buzzer(void)
{
if (fd >= 0) {
ioctl(fd, PWM_IOCTL_STOP);
close(fd);
```

```
fd = -1;
}
}
static void set_buzzer_freq(int freq)
{
// this IOCTL command is the key to set frequency
int ret = ioctl(fd, PWM_IOCTL_SET_FREQ, freq);
if(ret < 0) {
perror("set the frequency of the buzzer");
exit(1);
}
}
static void stop_buzzer(void)
{
int ret = ioctl(fd, PWM_IOCTL_STOP);
if(ret < 0) {
perror("stop the buzzer");
exit(1);
}
}
int main(int argc, char **argv)
{
int freq = 1000 ;
open_buzzer();
printf( "\nBUZZER TEST ( PWM Control )\n" );
printf( "Press +/- to increase/reduce the frequency of the BUZZER\n" ) ;
printf( "Press 'ESC' key to Exit this program\n\n" );
while( 1 )
{
int key;
set_buzzer_freq(freq);
printf( "\tFreq = %d\n", freq );
key = getch();
switch(key) {
case '+':
if( freq < 20000 )
freq += 10;
break;
case '-':
if( freq > 11 )
freq -= 10 ;
```

```
break;
case ESC_KEY:
case EOF:
stop_buzzer();
exit(0);
default:
break;
}
}
}
```

You can compile the program, download it and run

5.2.4 I2C-EEPROM Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/i2c/busses
Driver	I2c-s3c2410c
Device Type	Char
Device Name	/dev/i2c/0
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/i2c
Test Program Name	eeprog.c 24cxx.c
Executable Name	I2c
Test Program's Location in Board	
Note: the i2c driver has been compiled into the kernel by default and you cannot load it via insmod	
Program:	

```
Note: the following program depends on "24cxx.c" in the same directory.
#include <stdio.h>
#include <fcntl.h>
#include <getopt.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "24cXX.h"
#define usage_if(a) do { do_usage_if( a , __LINE__); } while(0);
void do_usage_if(int b, int line)
{
```

```
const static char *eeprog_usage =
"I2C-24C08(256 bytes) Read/Write Program, ONLY FOR TEST!\n"
"FriendlyARM Computer Tech. 2009\n";
if(!b)
return;
fprintf(stderr, "%s\n[line %d]\n", eeprog_usage, line);
exit(1);
}
#define die_if(a, msg) do { do_die_if( a , msg, __LINE__); } while(0);
void do_die_if(int b, char* msg, int line)
{
if(!b)
return;
fprintf(stderr, "Error at line %d: %s\n", line, msg);
fprintf(stderr, " sysmsg: %s\n", strerror(errno));
exit(1);
}
static int read_from_eeprom(struct eeprom *e, int addr, int size)
{
int ch, i;
for(i = 0; i < size; ++i, ++addr)
{
die_if((ch = eeprom_read_byte(e, addr)) < 0, "read error");
if( (i % 16) == 0 )
printf("\n %.4x| ", addr);
else if( (i % 8) == 0 )
printf(" ");
printf("%.2x ", ch);
fflush(stdout);
}
fprintf(stderr, "\n\n");
return 0;
}
static int write_to_eeprom(struct eeprom *e, int addr)
{
int i;
for(i=0, addr=0; i<256; i++, addr++)
{
if( (i % 16) == 0 )
printf("\n %.4x| ", addr);
else if( (i % 8) == 0 )
```

```
printf(" ");
printf("%.2x ", i);
fflush(stdout);
die_if(eeprom_write_byte(e, addr, i), "write error");
}
fprintf(stderr, "\n\n");
return 0;
}
int main(int argc, char** argv)
{
struct eeprom e;
int op;
op = 0;
usage_if(argc != 2 || argv[1][0] != '-' || argv[1][2] != '\0');
op = argv[1][1];
fprintf(stderr, "Open /dev/i2c/0 with 8bit mode\n");
die_if(eeprom_open("/dev/i2c/0", 0x50, EEPROM_TYPE_8BIT_ADDR, &e) < 0,
"unable to open eeprom device file "
"(check that the file exists and that it's readable)");
switch(op)
{
case 'r':
fprintf(stderr, " Reading 256 bytes from 0x0\n");
read_from_eeprom(&e, 0, 256);
break;
case 'w':
fprintf(stderr, " Writing 0x00-0xff into 24C08 \n");
write_to_eeprom(&e, 0);
break;
default:
usage_if(1);
exit(1);
}
eeprom_close(&e);
return 0;
}
```

You can compile the program, download it and run

5.2.5 Serial Port Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6drivers/serial/
Driver	S3c2451.c
Device Type	
Device Name	/dev/ttySAC0, 1, 2
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/comtest
Test Program Name	comtest.c
Executable Name	Armcomtest
Test Program's Location in Board	

Note: you can get two versions , one for x86 and the other for ARM. Both are generated from the same source code.

The comtest utility is developed by Friendly ARM to test serial ports. It is very similar to "minicom" in Linux and independent of hardware. Its source code can be applied in both any arm-linux platforms and PCs. This program is developed by Friendly ARM and unauthorized usage of it is forbidden

Program:

```
# include <stdio.h>
# include <stdlib.h>
# include <termio.h>
# include <unistd.h>
# include <fcntl.h>
# include <getopt.h>
# include <time.h>
# include <errno.h>
# include <string.h>
static void Error(const char *Msg)
{
fprintf (stderr, "%s\n", Msg);
fprintf (stderr, "strerror() is %s\n", strerror(errno));
exit(1);
}
static void Warning(const char *Msg)
{
fprintf (stderr, "Warning: %s\n", Msg);
}
static int SerialSpeed(const char *SpeedString)
{
int SpeedNumber = atoi(SpeedString);
```

```
# define TestSpeed(Speed) if (SpeedNumber == Speed) return B##Speed
TestSpeed(1200);
TestSpeed(2400);
TestSpeed(4800);
TestSpeed(9600);
TestSpeed(19200);
TestSpeed(38400);
TestSpeed(57600);
TestSpeed(115200);
TestSpeed(230400);
Error("Bad speed");
return -1;
}
static void PrintUsage(void)
{
fprintf(stderr, "comtest - interactive program of comm port\n");
fprintf(stderr, "press [ESC] 3 times to quit\n\n");
fprintf(stderr, "Usage: comtest [-d device] [-t tty] [-s speed] [-7] [-c] [-x] [-o]
[-h]\n");
fprintf(stderr, " -7 7 bit\n");
fprintf(stderr, " -x hex mode\n");
fprintf(stderr, " -o output to stdout too\n");
fprintf(stderr, " -c stdout output use color\n");
fprintf(stderr, " -h print this help\n");
exit(-1);
}
static inline void WaitFdWriteable(int Fd)
{
fd_set WriteSetFD;
FD_ZERO(&WriteSetFD);
FD_SET(Fd, &WriteSetFD);
if (select(Fd + 1, NULL, &WriteSetFD, NULL, NULL) < 0) {
Error(strerror(errno));
}
}
int main(int argc, char **argv)
{
int CommFd, TtyFd;
struct termios TtyAttr;
struct termios BackupTtyAttr;
int DeviceSpeed = B115200;
```

```
int TtySpeed = B115200;
int ByteBits = CS8;
const char *DeviceName = "/dev/ttyS0";
const char *TtyName = "/dev/tty";
int OutputHex = 0;
int OutputToStdout = 0;
int UseColor = 0;
opterr = 0;
for (;;) {
int c = getopt(argc, argv, "d:s:t:7xoch");
if (c == -1)
break;
switch(c) {
case 'd':
DeviceName = optarg;
break;
case 't':
TtyName = optarg;
break;
case 's':
if (optarg[0] == 'd') {
DeviceSpeed = SerialSpeed(optarg + 1);
} else if (optarg[0] == 't') {
TtySpeed = SerialSpeed(optarg + 1);
} else
TtySpeed = DeviceSpeed = SerialSpeed(optarg);
break;
case 'o':
OutputToStdout = 1;
break;
case '7':
ByteBits = CS7;
break;
case 'x':
OutputHex = 1;
break;
case 'c':
UseColor = 1;
break;
case '?':
case 'h':
```

```
default:
PrintUsage();
}
}
if (optind != argc)
PrintUsage();
CommFd = open(DeviceName, O_RDWR, 0);
if (CommFd < 0)
Error("Unable to open device");
if (fcntl(CommFd, F_SETFL, O_NONBLOCK) < 0)
Error("Unable set to NONBLOCK mode");
memset(&TtyAttr, 0, sizeof(struct termios));
TtyAttr.c_iflag = IGNPAR;
TtyAttr.c_cflag = DeviceSpeed | HUPCL | ByteBits | CREAD | CLOCAL;
TtyAttr.c_cc[VMIN] = 1;
if (tcsetattr(CommFd, TCSANOW, &TtyAttr) < 0)
Warning("Unable to set comm port");
TtyFd = open(TtyName, O_RDWR | O_NDELAY, 0);
if (TtyFd < 0)
Error("Unable to open tty");
TtyAttr.c_cflag = TtySpeed | HUPCL | ByteBits | CREAD | CLOCAL;
if (tcgetattr(TtyFd, &BackupTtyAttr) < 0)
Error("Unable to get tty");
if (tcsetattr(TtyFd, TCSANOW, &TtyAttr) < 0)
Error("Unable to set tty");
for (;;) {
unsigned char Char = 0;
fd_set ReadSetFD;
void OutputStdChar(FILE *File) {
char Buffer[10];
int Len = sprintf(Buffer, OutputHex ? "%.2X " : "%c", Char);
fwrite(Buffer, 1, Len, File);
}
FD_ZERO(&ReadSetFD);
FD_SET(CommFd, &ReadSetFD);
FD_SET( TtyFd, &ReadSetFD);
# define max(x,y) ( ((x) >= (y)) ? (x) : (y) )
if (select(max(CommFd, TtyFd) + 1, &ReadSetFD, NULL, NULL, NULL) < 0) {
Error(strerror(errno));
}
# undef max
```

```
if (FD_ISSET(CommFd, &ReadSetFD)) {
while (read(CommFd, &Char, 1) == 1) {
WaitFdWriteable(TtyFd);
if (write(TtyFd, &Char, 1) < 0) {
Error(strerror(errno));
}
if (OutputToStdout) {
if (UseColor)
fwrite("\x1b[01;34m", 1, 8, stdout);
OutputStdChar(stdout);
if (UseColor)
fwrite("\x1b[00m", 1, 8, stdout);
fflush(stdout);
}
}
}
if (FD_ISSET(TtyFd, &ReadSetFD)) {
while (read(TtyFd, &Char, 1) == 1) {
static int EscKeyCount = 0;
WaitFdWriteable(CommFd);
if (write(CommFd, &Char, 1) < 0) {
Error(strerror(errno));
}
if (OutputToStdout) {
if (UseColor)
fwrite("\x1b[01;31m", 1, 8, stderr);
OutputStdChar(stderr);
if (UseColor)
fwrite("\x1b[00m", 1, 8, stderr);
fflush(stderr);
}
if (Char == '\x1b') {
EscKeyCount ++;
if (EscKeyCount >= 3)
goto ExitLabel;
} else
EscKeyCount = 0;
}
}
}
ExitLabel:
```

```
if (tcsetattr(TtyFd, TCSANOW, &BackupTtyAttr) < 0)
Error("Unable to set tty");
return 0;
}
```

You can compile the program, download it and run

5.2.6 UDP Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/net/
Driver	dm9000.c
Device Type	
Device Name	eth0 (not listed in /dev)
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/udptak
Test Program Name	udptalk.c
Executable Name	udptalk
Test Program's Location in Board	

Program:

```
/*
* udptalk : Example for Matrix V ;this program applies to the mini2451 system too
*
* Copyright (C) 2004 capbily - friendly-arm
* capbily@hotmail.com
*/
#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, /*remote IP and socket socket address*/
localaddr;/*Local socket address*/
int sockfd;
char recmsg[BUFLEN+1];
int socklen, n;
if(argc!=5){
printf("%s <dest IP address> <dest port> <source IP address> <source port>\n", argv[0]);
```

```
exit(0);
}
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if(sockfd<0){
printf("socket creating err in udptalk\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 dest 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 source IP address!\n");
exit(0);
}
localaddr.sin_port=htons(atoi(argv[4]));
if(bind(sockfd, &localaddr, socklen)<0){
printf("bind local address err in udptalk!\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 udptalk!\n");
exit(3);
}
for(;;){
/*recv&send message loop*/
n = recvfrom(sockfd, recmsg, BUFLEN, 0, &peeraddr, &socklen);
if(n<0){
printf("recvfrom err in udptalk!\n");
exit(4);
}else{
/*received data*/
recmsg[n]=0;
printf("peer:%s", recmsg);
```

```
}
if(fgets(recmsg, BUFLEN, stdin) == NULL) exit(0);
if(sendto(sockfd, recmsg, strlen(recmsg), 0, &peeraddr, socklen)<0){
printf("sendto err in udptalk!\n");
exit(3);
}
}
}
```

Test:

Please compile “udptalk.c”. There are two executables under “/opt/FriendlyARM/mini2451/examples/udptalk”, one x86-udptalk and the other arm-udptalk. The make command will generate both. Please download “arm-udptalk” to the board (the preinstalled Linux doesn’t have this). In our example the host IP is 192.168.1.108 and the board’s IP is 192.168.1.230.

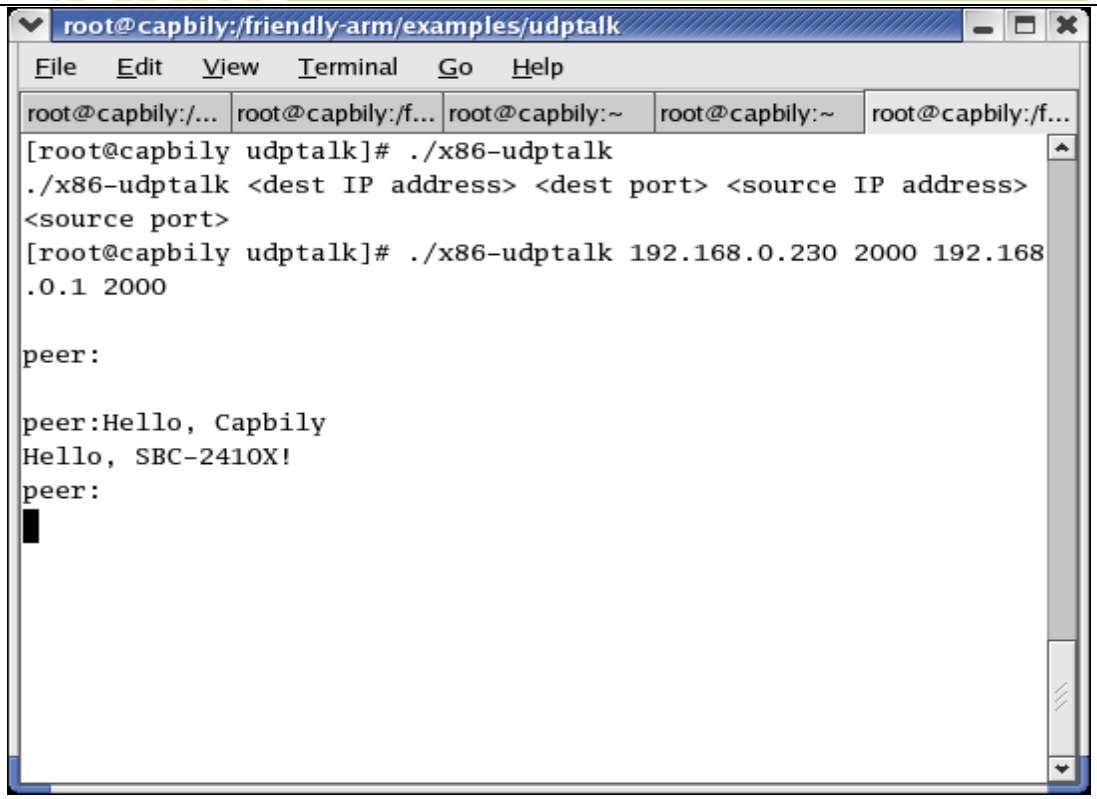
Type the following command on your host:

#./x86-udptalk 192.168.1.230 2000 192.168.1.108 2000

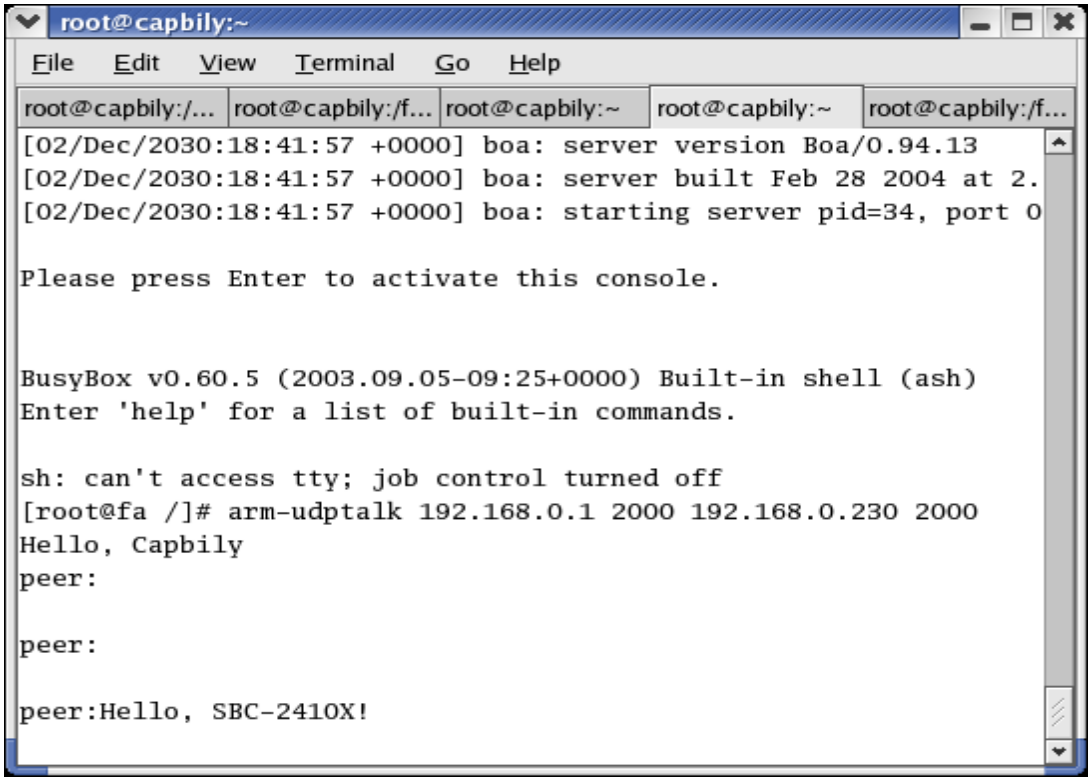
Type the following command on your board

#arm-udptalk 192.168.1.108 2000 192.168.1.230 2000

You will see the following results:



x86-udptalk running on host



arm-udptalk running on board

5.2.7 Utiliz Math Libraries

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/math
Test Program Name	mathtest.c
Executable Name	Mathtest
Test Program's Location in Board	

Note: to utilize math libraries you need to include its header file "**math.h**" and add an compile option **libm**

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h> ; note: including this header file is a must
int main(void)
{
double a=8.733243;
printf("sqrt(%f)=%f\n", a, sqrt(a));
return 0;
}


Makefile:
CROSS=arm-linux-
all: mathtest
#It includes the math library "libm", marked in red
mathtest:
$(CROSS)gcc -o mathtest main.c -lm
clean:
@rm -vf mathtest *.o *~
```

5.2.8 Thread Programming

Program Description:	
Source Code Location	

Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/pthread
Test Program Name	pthread_test.c
Executable Name	pthread_test
Test Program's Location in Board	

Note: to utilize math libraries you need to include its header file "pthread.h" and add an compile option libpthread

Program:

```
#include<stddef.h>
#include<stdio.h>
#include<unistd.h>
#include"pthread.h" ; including this header is a must
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='a';
printf("make a new item\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("consume item\n");
buffer_has_item=0;
}
pthread_mutex_unlock(&mutex);
}
}
Makefile:
CROSS=arm-linux-
all: pthread
#note: it includes the thread library libphread marked in red
pthread:
$(CROSS)gcc -static -o pthread main.c -lpthread
clean:
@rm -vf pthread *.o *~
```

5.2.9 Pipe Programming – Manipulating LED Remotely

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/led-player
Test Program Name	led-player.c
Executable Name	led-player
Test Program's Location in Board	
Note: to utilize math libraries you need to include its header file "pthread.h" and add an compile option libpthread	
Program:	
#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <sys/ioctl.h>	

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <string.h>
static int led_fd;
static int type = 1;
static void push_leds(void)
{
static unsigned step;
unsigned led_bitmap;
int i;
switch(type) {
case 0:
if (step >= 6) {
step = 0;
}
if (step < 3) {
led_bitmap = 1 << step;
} else {
led_bitmap = 1 << (6 - step);
}
break;
case 1:
if (step > 255) {
step = 0;
}
led_bitmap = step;
break;
default:
led_bitmap = 0;
}
step++;
for (i = 0; i < 4; i++) {
ioctl(led_fd, led_bitmap & 1, i);
led_bitmap >>= 1;
}
}
int main(void)
{
```

```
int led_control_pipe;
int null_writer_fd; // for read endpoint not blocking when control process exit
double period = 0.5;
led_fd = open("/dev/leds0", 0);
if (led_fd < 0) {
led_fd = open("/dev/leds", 0);
}
if (led_fd < 0) {
perror("open device leds");
exit(1);
}
unlink("/tmp/led-control");
mkfifo("/tmp/led-control", 0666);
led_control_pipe = open("/tmp/led-control", O_RDONLY | O_NONBLOCK);
if (led_control_pipe < 0) {
perror("open control pipe for read");
exit(1);
}
null_writer_fd = open("/tmp/led-control", O_WRONLY | O_NONBLOCK);
if (null_writer_fd < 0) {
perror("open control pipe for write");
exit(1);
}
for (;;) {
fd_set rds;
struct timeval step;
int ret;
FD_ZERO(&rds);
FD_SET(led_control_pipe, &rds);
step.tv_sec = period;
step.tv_usec = (period - step.tv_sec) * 1000000L;
ret = select(led_control_pipe + 1, &rds, NULL, NULL, &step);
if (ret < 0) {
perror("select");
exit(1);
}
if (ret == 0) {
push_leds();
} else if (FD_ISSET(led_control_pipe, &rds)) {
static char buffer[200];
for (;;) {
```

```
char c;
int len = strlen(buffer);
if (len >= sizeof buffer - 1) {
memset(buffer, 0, sizeof buffer);
break;
}
if (read(led_control_pipe, &c, 1) != 1) {
break;
}
if (c == '\r') {
continue;
}
if (c == '\n') {
int tmp_type;
double tmp_period;
if (sscanf(buffer,"%d%lf", &tmp_type, &tmp_period) == 2) {
type = tmp_type;
period = tmp_period;
}
fprintf(stderr, "type is %d, period is %lf\n", type, period);
memset(buffer, 0, sizeof buffer);
break;
}
buffer[len] = c;
}
}
}
close(led_fd);
return 0;
}
```

"make" will generate a led-player executable which is run as a server under "/sbin". The leds.cgi gateway source code is under "/www/leds.cgi" on the board. It is a shell script and can be invoked by leds.html as an action. Here is the shell file:

leds.cgi:

```
#!/bin/sh
type=0
period=1
case $QUERY_STRING in
*ping*)
type=0
;;
*counter*)
type=1
;;
*stop*)
type=2
;;
esac
case $QUERY_STRING in
*slow*)
period=0.25
;;
*normal*)
period=0.125
;;
*fast*)
period=0.0625
;;
esac
/bin/echo $type $period > /tmp/led-control
echo "Content-type: text/html; charset=gb2312"
echo
/bin/cat led-result.template
exit 0
```

5.2.10 “Hello World” with C++

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/C++
Test Program Name	cplus.c

Executable Name	cplus
Test Program's Location in Board	

Note: to utilize math libraries you need to include its header file "pthread.h" and add an compile option libpthread

Program:

```
#include <iostream>
#include <cstring>
using namespace std;
class String
{
private:
char *str;
public:
String(char *s)
{
int lenght=strlen(s);
str = new char[lenght+1];
strcpy(str, s);
}
~String()
{
cout << "Deleting str.\n";
delete[] str;
}
void display()
{
cout << str <<endl;
}
};
int main(void)
{
String s1="I like FriendlyARM.";
cout << "s1=";
s1.display();
return 0;
double num, ans;
cout << "Enter num:";
}
```

5.3 Sample Linux Module

The “Hello,World” introduced in the previous section runs in user mode. Now we will present a program “Hello, World” that runs in kernel mode and take this as an example to show you how to write a driver

5.3.1 Souce Code of Hello Module

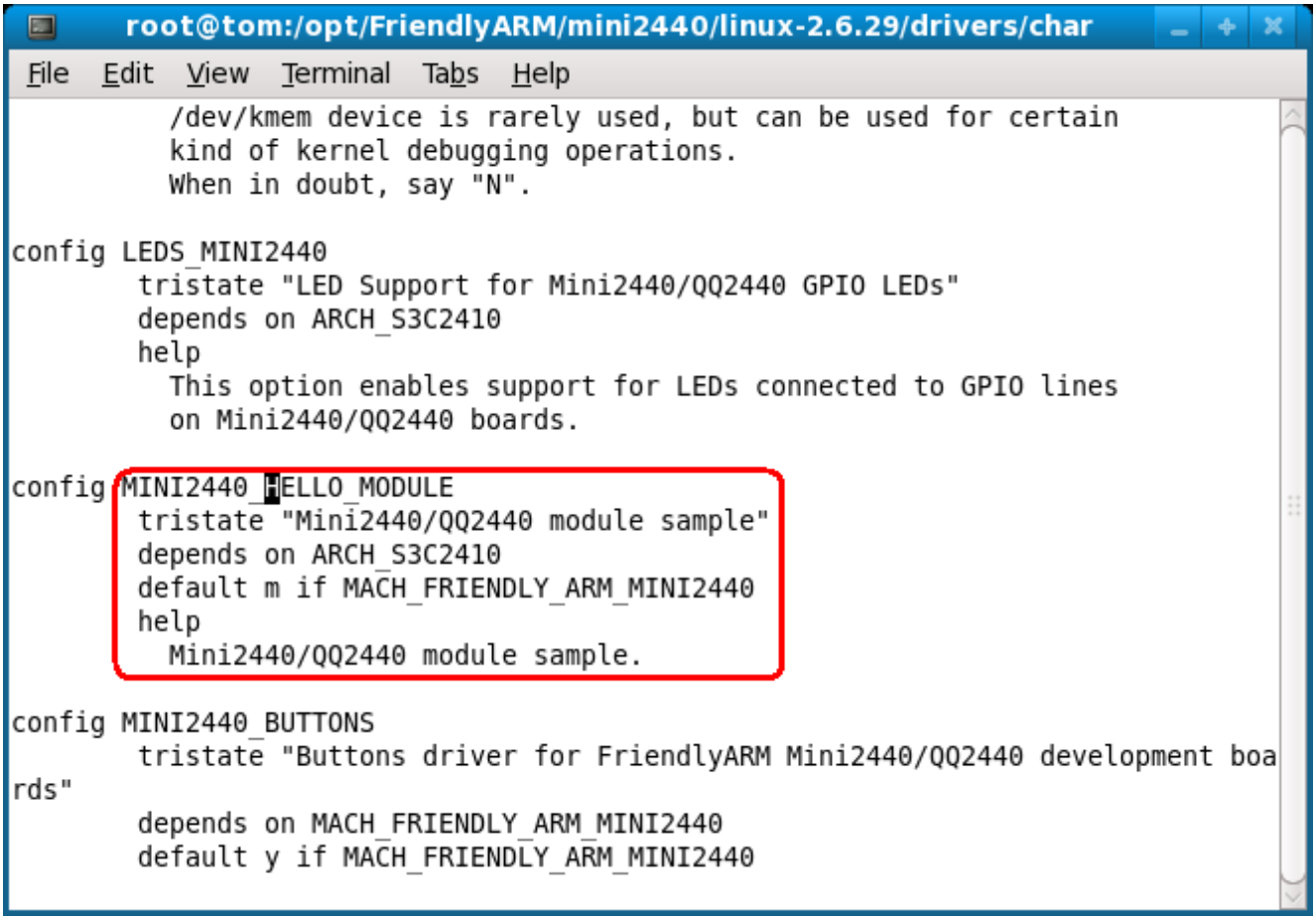
Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char
Test Program Name	Mini2451_hello_module.c
Executable Name	
Test Program’s Location in Board	
Note: mounting this driver will not create a device node under /dev	
Program:	

```
#include <linux/kernel.h>
#include <linux/module.h>
static int __init mini2451_hello_module_init(void)
{
printk("Hello, Mini2451 module is installed !\n");
return 0;
}
static void __exit mini2451_hello_module_cleanup(void)
{
printk("Good-bye, Mini2451 module was removed!\n");
}
module_init(mini2451_hello_module_init);
module_exit(mini2451_hello_module_cleanup);
MODULE_LICENSE("GPL");
```

5.3.2 Install Hello Module

Please follow the steps below to include the module into the kernel and compile (Note: actually the following steps have been set up and you only need to directly compile it)：

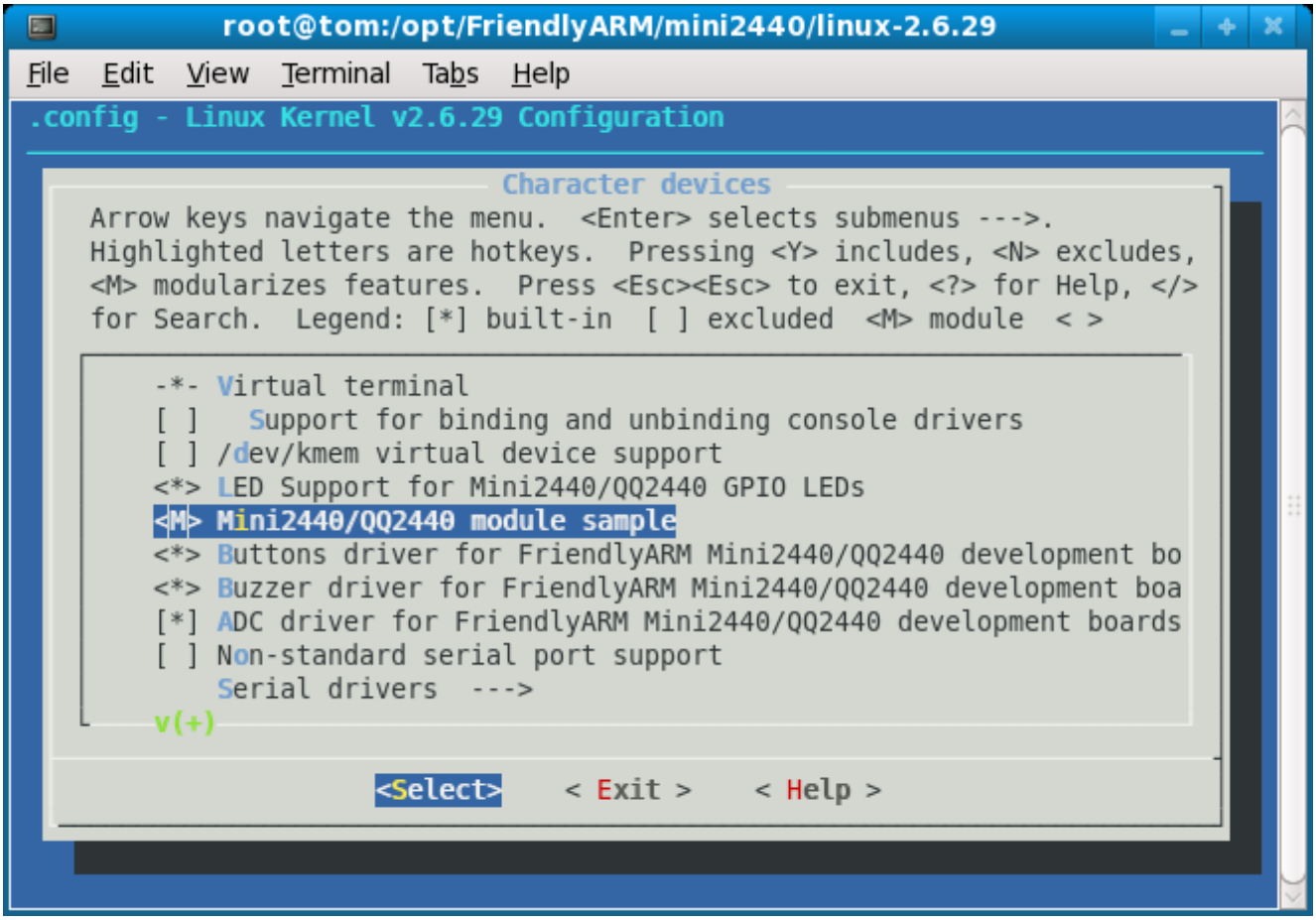
Step1: configure “Kconfig”, add this module in the drivers and it will appear in make menuconfig. Open “linux-2.6.32.2/drivers/char/Kconfig” add lined marked in yellow



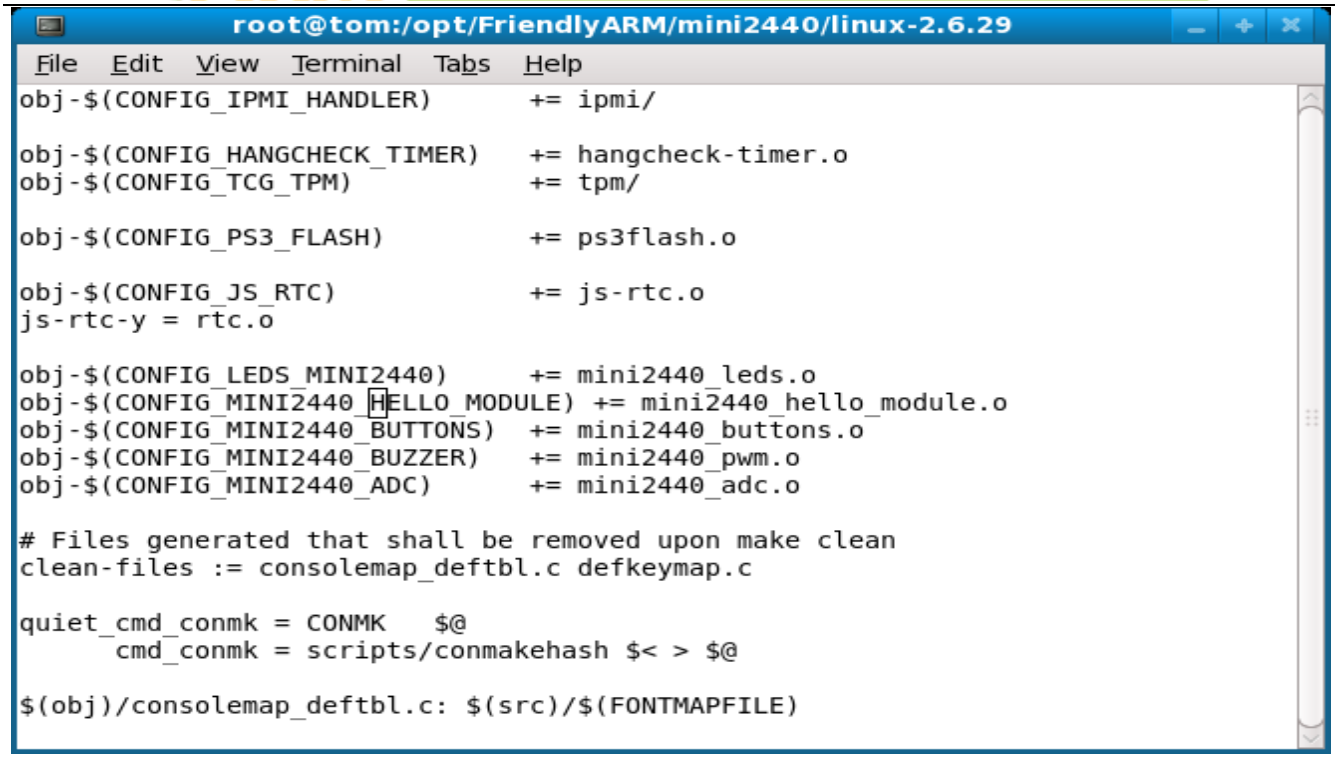
Save and exit. When you run “make menuconfig” in the linux-2.6.32.2 directory you will see your item in Device Drivers -> Character devices. Press the space key it will be marked “<M>”. This means this souce code will be compiled as a module. Press the

space key again it will be marked “<*>”. This means it will be compiled into the kernel.

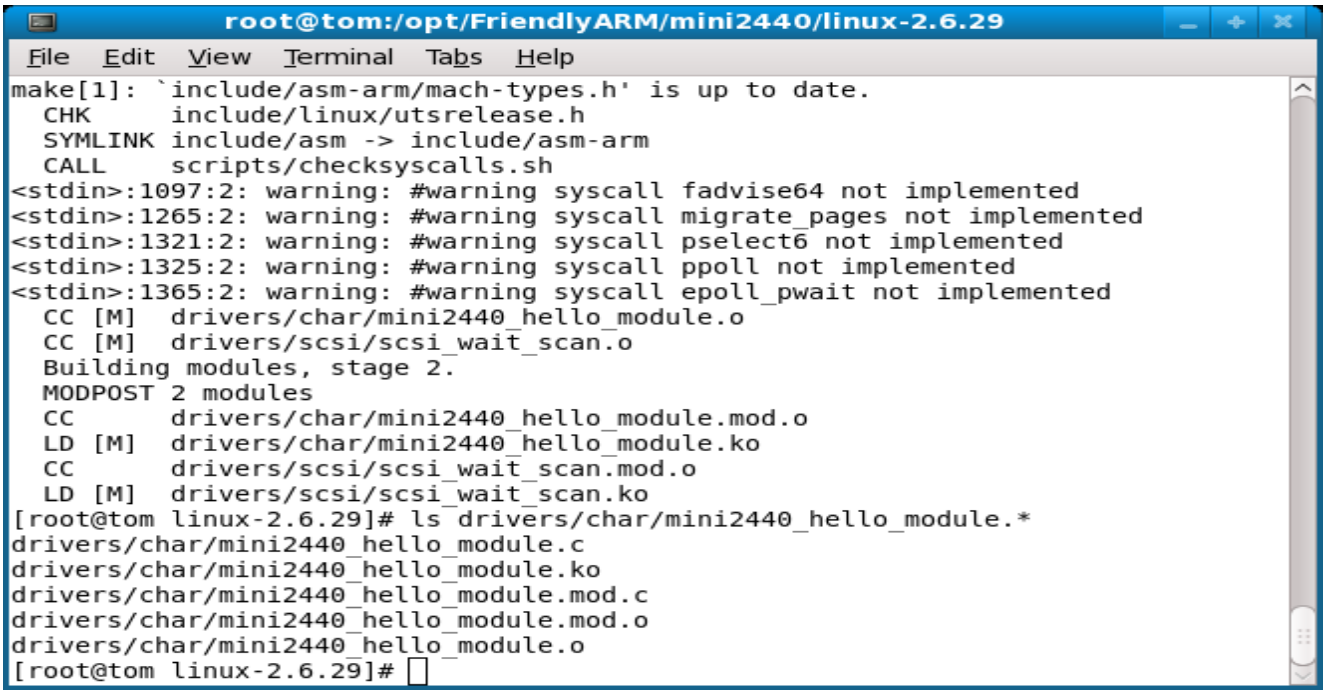
Here we chose “<M>”



Step2: the previous step still cannot include it into the kernel when compiling. You need to linkthe kernel configuration to the source code in “makefile”. Open “linux-2.6.32.2/drivers/char/Makefile”, add the marked line shown below, save and exit



Step3: go back to the linux-2.6.32.2 source code directory, run “make modules” a “mini2451_hello_module.ko” module will be generated. Prior to executing “make modules”, you need to run “make zImage”. This only needs to be run once.



5.3.3 Download Hello Module

Please transfer "mini2451_hello_module.ko" to the board via FTP and move it to "**/lib/modules/3.6.0-FriendlyARM**"

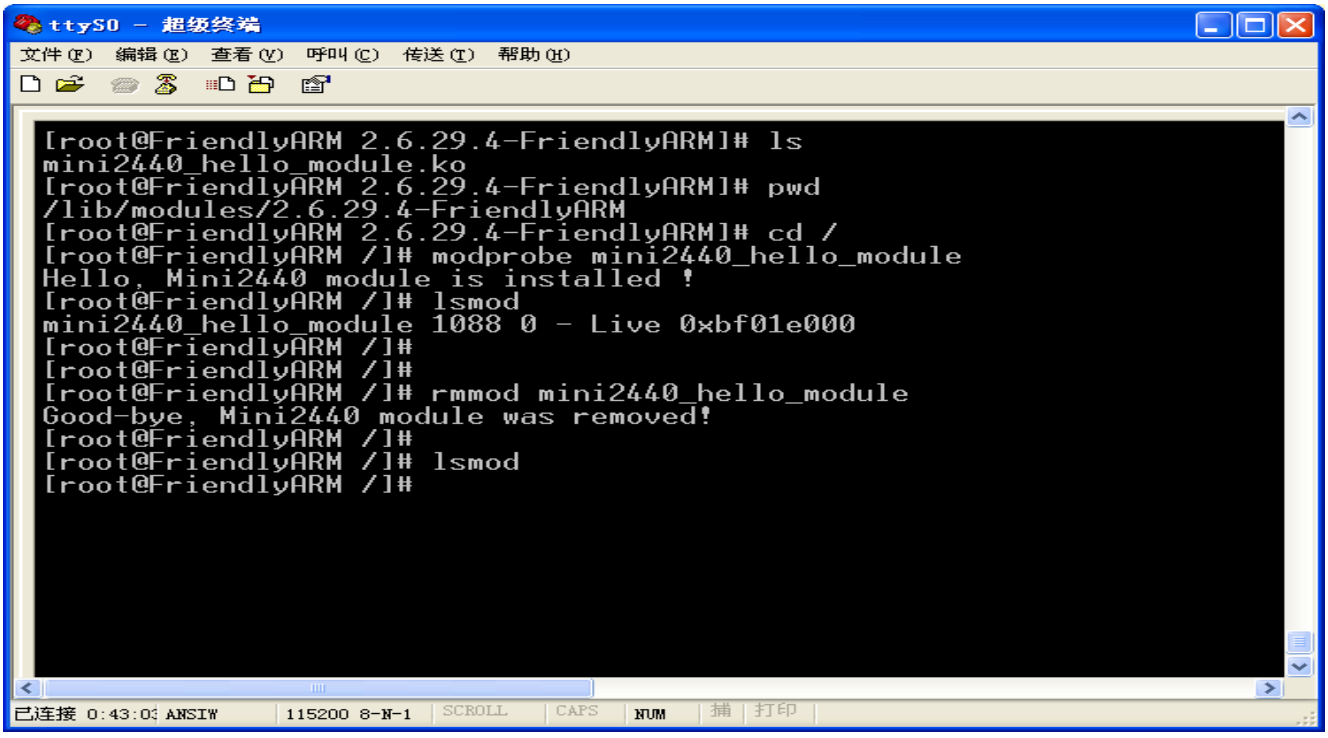
#modprobe mini2451_hello_module

You can observe that the module has been loaded (note: to load a module with "modprobe" you don't need to add the "ko" extension)

Run the following command you will observe that the module has been unmounted

#rmmod mini2451_hello_module

Note: to load a module correctly, you need to move your module to the boards's "**/lib/modules/3.6.0-FriendlyARM**" directory. In addition, if your kernel's version is different from the example here please create a new directory for your kernel. Here it is **/lib/modules/3.6.0-FriendlyARM.**



5.4 Sample Linux Driver

5.4.1 LED Driver

In this example we will present an LED driver program which can drive the 4 LEDs on the board

LED	IO Register	CPU Pin
LED1	GPB5	K2
LED2	GPB6	L5
LED3	GPB7	K7
LED4	GPB8	K5

To manipulate an IO you need to set up its register by invoking some functions and macros. Here we used “readl” and “writel”. They can directly read and write corresponding registers. Besides you need some other driver related functions too such as misc_register, module_init, module_exit and filling the file_operations structure.

Program Description:	
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char
Driver	Mini2451_leds.c
Device Type	Misc, auto generated
Device Name	/dev/leds
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/leds
Test Program Name	led.c
Executable Name	Led
Test Program’s Location in Board	
Note: the LED driver has been compiled into the kernel by default and cannot be loaded via insmod	
Program:	

5.4.2 Button Driver

Program Description:		
Source Code Location	/opt/FriendlyARM/mini2451/linux-3.6/drivers/char	
Driver	Mini2451_buttons.c	
Device Type	Misc, auto generated	
Device Name	/dev/buttons	
Test Program Source Code Location	/opt/FriendlyARM/mini2451/examples/buttons	
Test Program Name	buttons_test.c	
Executable Name	buttons	
Test Program's Location in Board		
Note: the button driver has been compiled into the kernel by default and cannot be loaded via insmod		
Key	IO	Interrupt
K1	GPG0	EINT8
K2	GPG3	EINT11
K3	GPG5	EINT13
K4	GPG6	EINT14
K5	GPG7	EINT15
K6	GPG11	EINT19
Program:		

5.5 Compile Qtopia-2.2.0

To make it easy for users we compile all the steps into one build script. Executing this script will compile thewhole qtopia platform and its utilities. You can start them by commanding "**run**". The compiling scripts for x86 and arm are a little bit different.

5.5.1 Uncompress and Install Source Code

Please refer to 3.4.1

5.5.2 Compile and Run Qtopia-2.2.0 for X86

All our programs have been verified on Fedora9. We didn't try them on other platforms. We strongly recommend our users to use Fedora9 and download it from ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/Fedora/i386/iso/Fedora-9-i386-DVD.iso.

Enter the working directory and run the following command

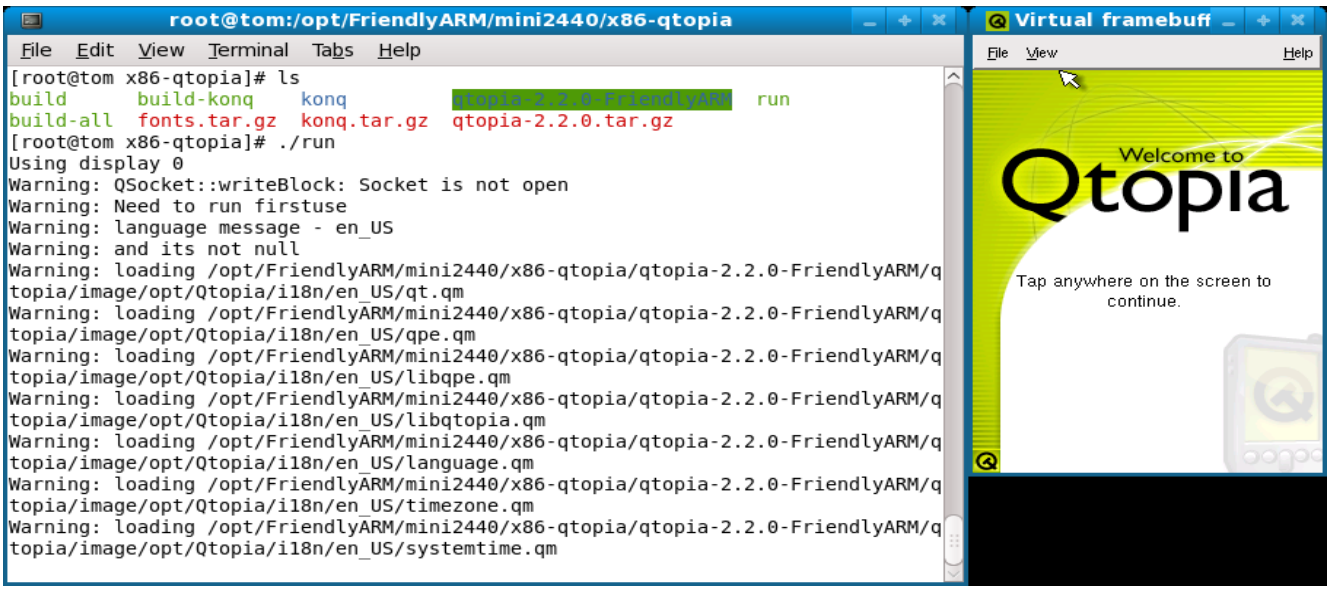
#**cd /opt/FriendlyARM/mini2451/x86-qtopia**

#**./build-all** (this process takes about 30 minutes)

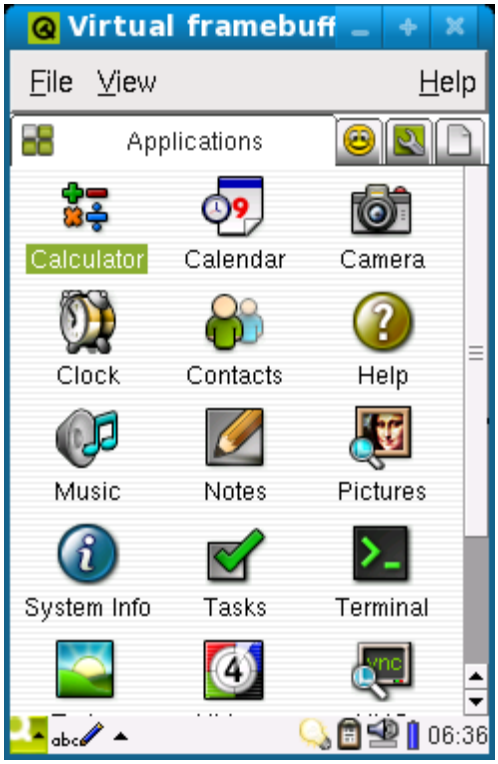
Note: **./build-all** will automatically compile the complete Qtopia and its embedded web browser. You can execute "**./build**" first and then "./**build-konq**" to compile them separately. To run your qtopia you can type the command below:

#**./run**

You will see the following screen



Follow the default options to continue and you will see the following screen



5.5.3 Compile and Run Qtopia-2.2.0 for ARM

Please make sure your compiler is arm-linux-gcc-4.4.3 and platform is Fedora 9.

Enter the working directory and type the command below

#**cd /opt/FriendlyARM/mini2451/arm-qtopia**

#**./build-all** (this process takes about 30 minutes)

#./mktarget (this makes a file system image and will generate

"**target-qtopia-konq.tgz**")

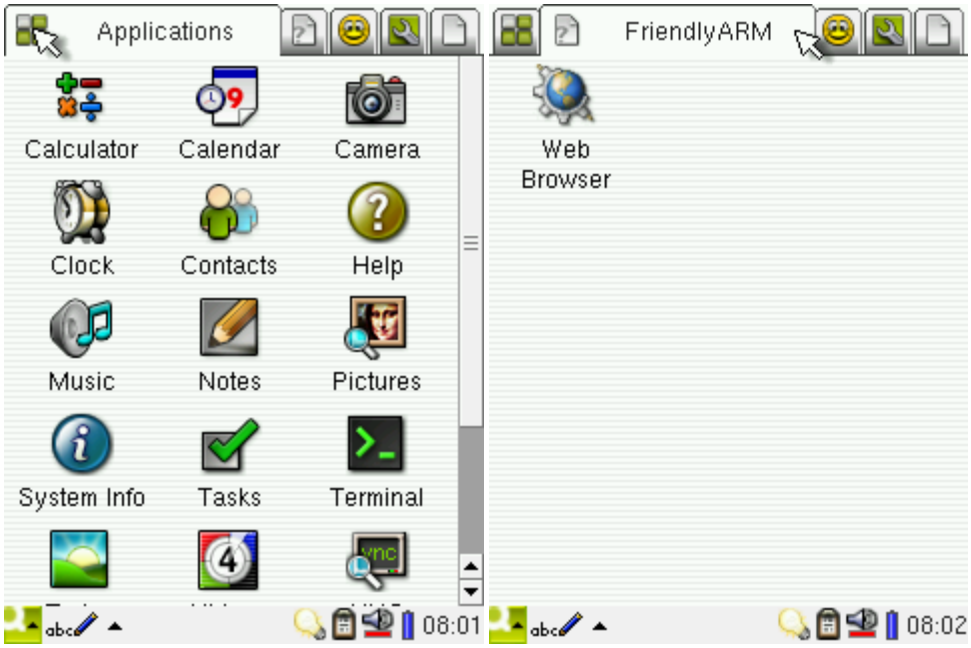
Note: "**./build-all**" will automatically compile a complete Qtopia system and the web browser and generate Jpeg, GIF, PNG image files. You can execute "**./build**" first and then "./**build-konq**" to compile them separately.

To remove your old Qtopia system you just need to delete all the files under "/opt". Then you can uncompress your target-qtopia-konq.tgz to the board's root directory via a flash drive. In our example we had it under /home/plg. Please run the command below:

#tar xvzf /home/plg/target-qtopia-konq.tgz –C /

"C" means "Change" and "/" after "C" means it will be uncompressed to the root directory. After you are done reboot your board and you will see that all your GUI components are in English now and there is a browser under the "FriendlyARM" tag. This is your own Qtopia.

Note: your new system may load parameters from "/etc/pointercal" you can delete that file too and will be directed to the calibration screen after reboot.



The above procedure is a simplied one. We hide all technical details in the build-all script you can look into it for more details

5.6 Compile QtE-4.8.5

5.6.1 Uncompress and Install Source Code

Please refer to 3.4.1

5.6.2 Compile and Run QtE-4.8.5 for ARM

Note: please use our arm-linux-gcc-4.4.3 and Fedora9 to compile. We offered a build-all script for users to easily compile QtE-4.8.5. Please enter the source code directory and type the following command:

#cd /opt/FriendlyARM/mini2451/arm-qte-4.8.5

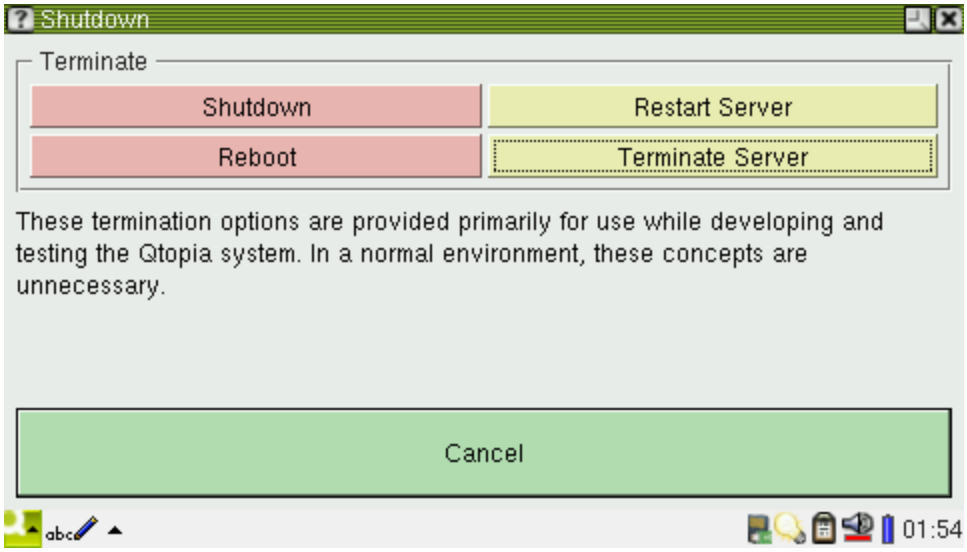
#./build.sh

The build process takes a while. And after it is done, please run the mktarget script and a **target-qte-4.8.5.tgz** will be generated. Please follow the command below:

#tar xvzf target-qte-4.8.5.tgz –C /

A Trolltech directory will be generated under "/usr/local/", which includes all needed libraries and executables. Since our shipped Linux already includes QtE-4.8.5, to test your build you can delete the one on your board by "rm" the whole "**/usr/local/Trolltech**" directory.

Before running QtE-4.8.5, please stop the current running Qtopia-2.2.0. Go to "Settings" -> "Shutdown" and you will see the following screen. Click on "Terminate Server" to shut down Qtopia-2.2.0.



Or you can shut it down: either by commenting out the qtopia option in the init script ”/etc/init.d/rcS” and rebooting the system or commanding “kill all” to terminate related process (there are many options: you can even delete the whole “/opt”, shut down qtopia-2.2.0 and run “qt4”