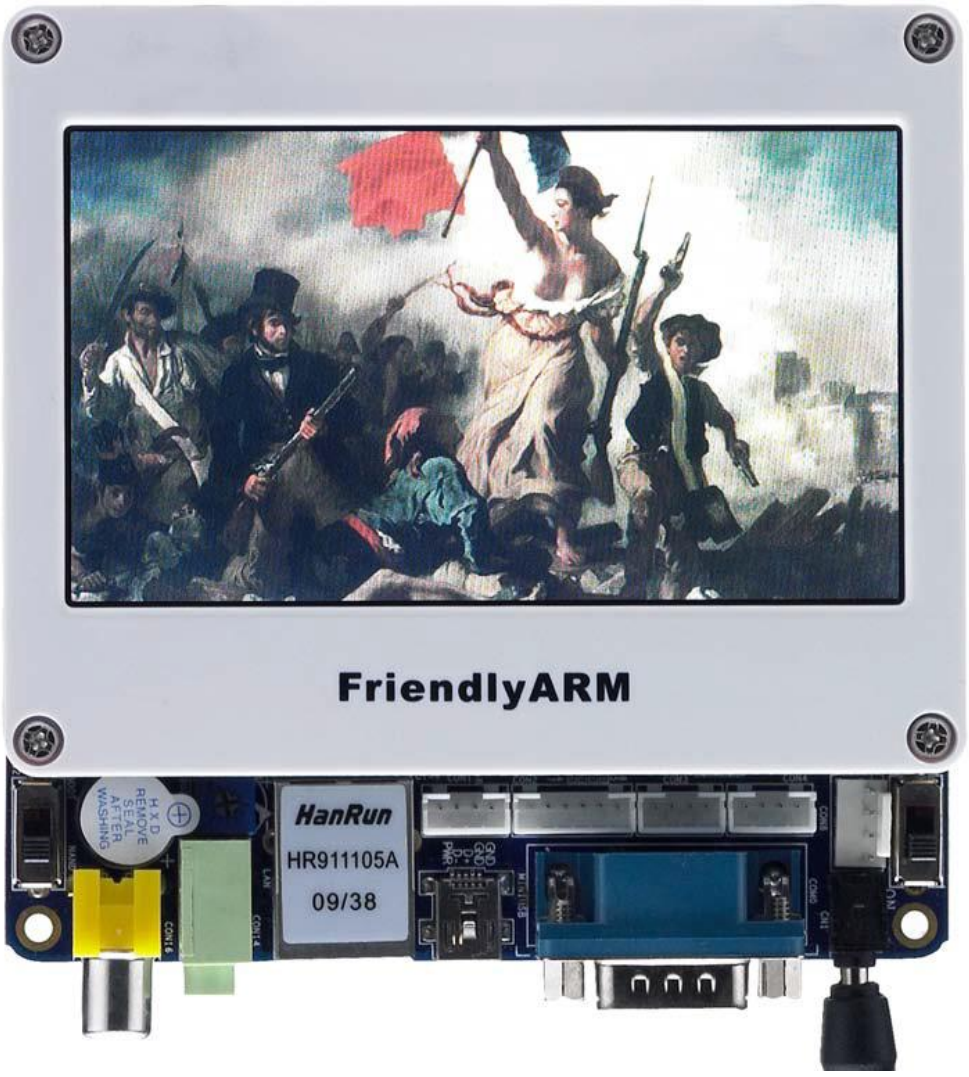
User's Guide to Mini6410 Linux



REVISION	ORIGINATOR	SCR	REV DATE
0.1.0	FriendlyARM Co., Ltd		March 28th, 2011
FriendlyARM 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 FriendlyARM Co., Ltd.			**FriendlyARM Co., Ltd** **Guangzhou, China**

Address: Room 1705,Block A1, Longyuan Plaza, 549# Longkouxi Road, Guangzhou, China, 510640
Website: http://www.arm9.net Email: capbily@163.com
Tel: +86-20-85201025 Fax: +86-20-85261505

COPYRIGHT STATEMENT

Index

Before step into this guide we recommend users to read our dummy book to obtain some handson experiences about the Mini6410 system.

For the sake of users we have made all our 6410 GUI utilities easy and simple to use. Applications like the camera preview, serial asistant, test button, screen rotation, and switching among Qtopia-2.2.0, Qtopia4 and QtE-4.7.0 are extremely user friendly. We would be glad to see our Mini6410 being your gateway to the wonderful world of embedded system development and learning.

1 Get Started with Qtopia-2.2.0, Qtopia4 and QtE-4.7.0

Note: Qtopia 2.2.0 is developed by Qt based on Qt/Embedded 2.3 graphic interface. After Qtopia 2.2.0, Qt hasn't released any new PDA versioned graphic interface. The latest Qtopia is for cell phones (Qt Extended 4.4.3). But it is still developing Qt/Embedded libaries.

To get the latest QtE please visit http://qt.nokia.com/. The version utilized in our system is QtE-4.7.0. Qt Extended 4.4.3 is the last version and we call it Qtopia4.

For details on how to compile Qtopia-2.2.0 please refer to 4.11

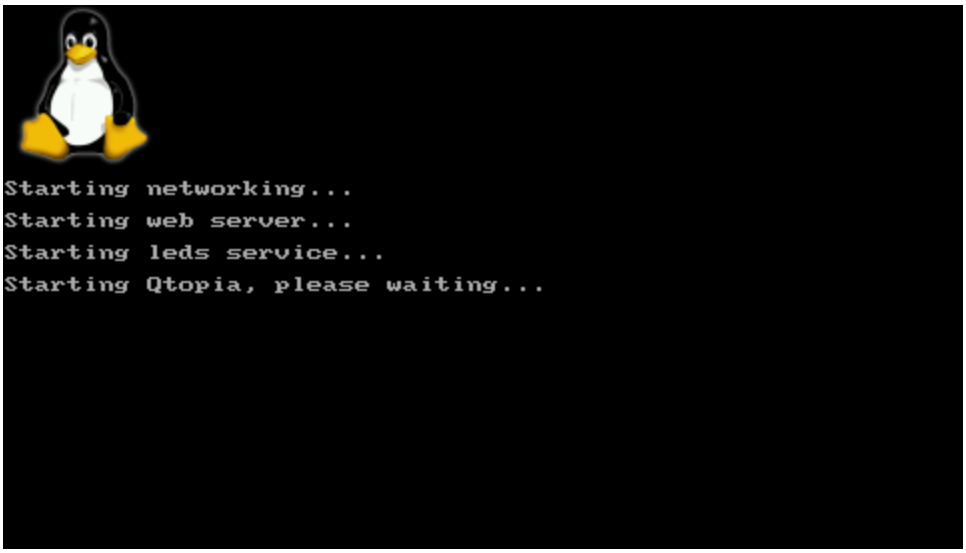
For details on how to compile QtE-4.7.0 please refer to 4.12

For details on how to compile Qtopia4 please refer to 4.13

For most of our released systems, we have installed Linux + Qtopia 2.2.0+Qtopia4+QtE-4.7.0+SMPlayer. It includes various useful utilities. Just power on your board you will be able to explore them.

Note: all the steps below are based on our 4.3"LCD system.

Let's go!

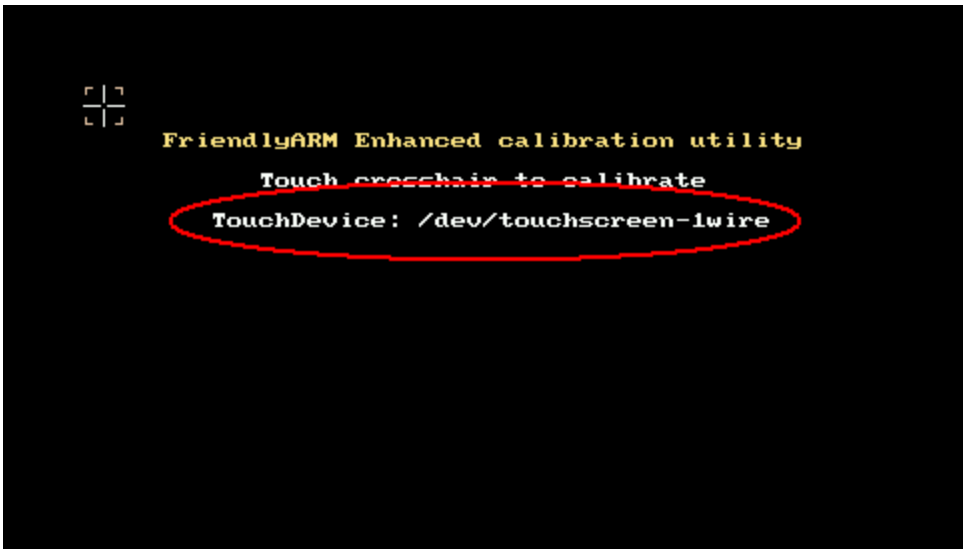


1.1 Calibrate Touch Screen

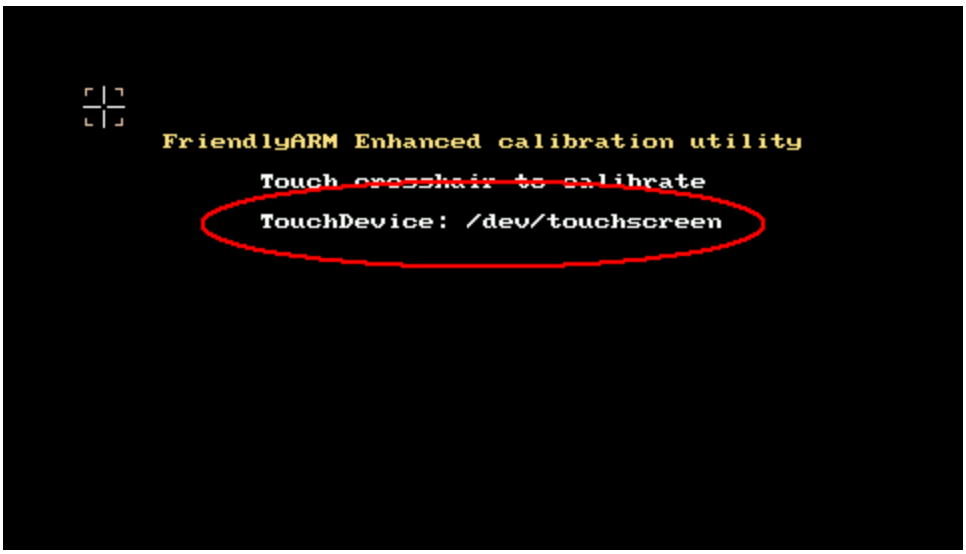
Note: if you cannot calibrate your screen by following the steps below, please delete "/etc/pointercal" and reboot or reinstall the whole system; or connect a USB mouse to your board, select "recalibrate" in "setting" to recalibrate your screen.

You will see the calibration interface under the following two situations:

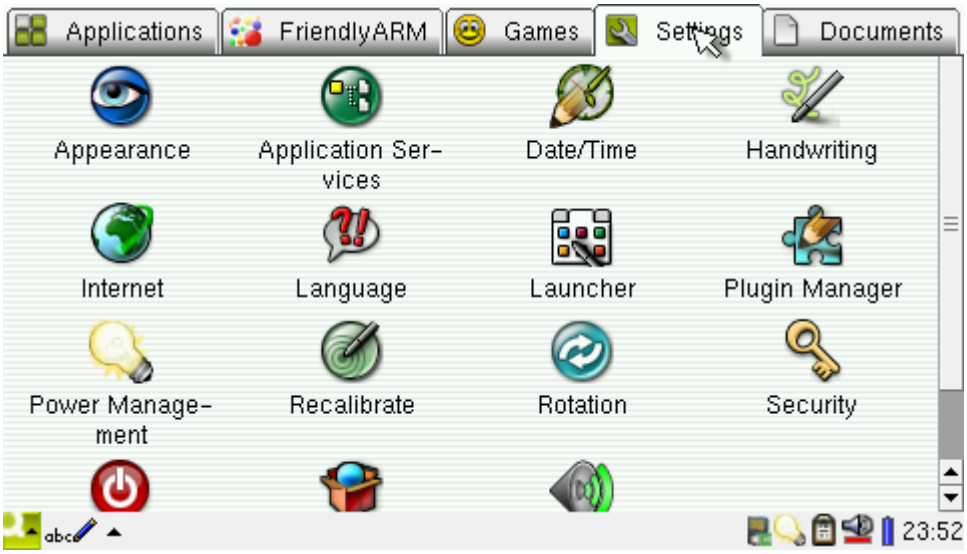
1. After you follow the steps to install the Qtopia system and reboot the system, you will see the screenshot below. Follow the prompts on the screen to click on them and then click on the "+" signals.



The statement red circled indicates that the system has the 1-wire precise touch device:/dev/touchscreen-1wire, if the ARM system has an integrated touch screen interface it will be "/dev/touchscreen"

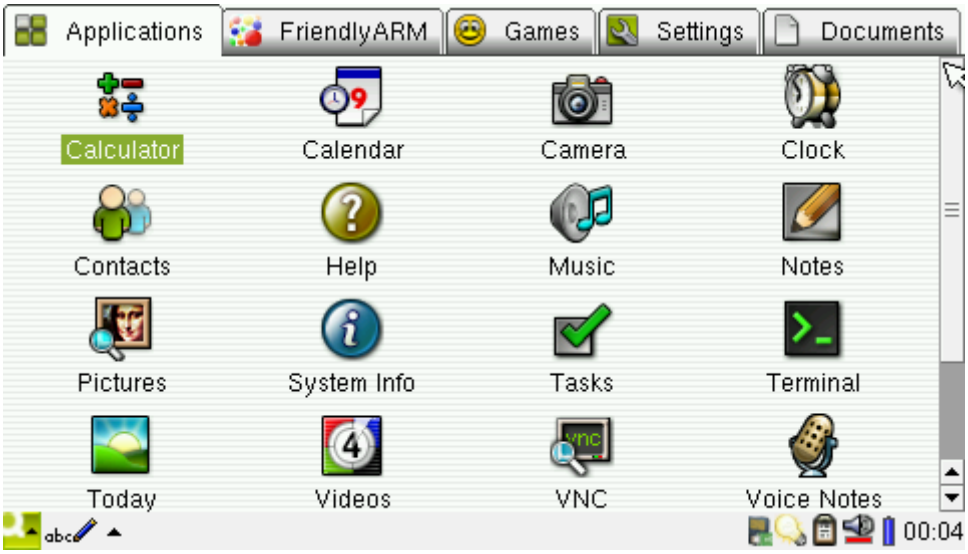


2. After entering the system, go to "Start" -> "Settings" -> "Configurations" -> "Recalibrate". Click on the "+" signal.



1.2 Main Interface

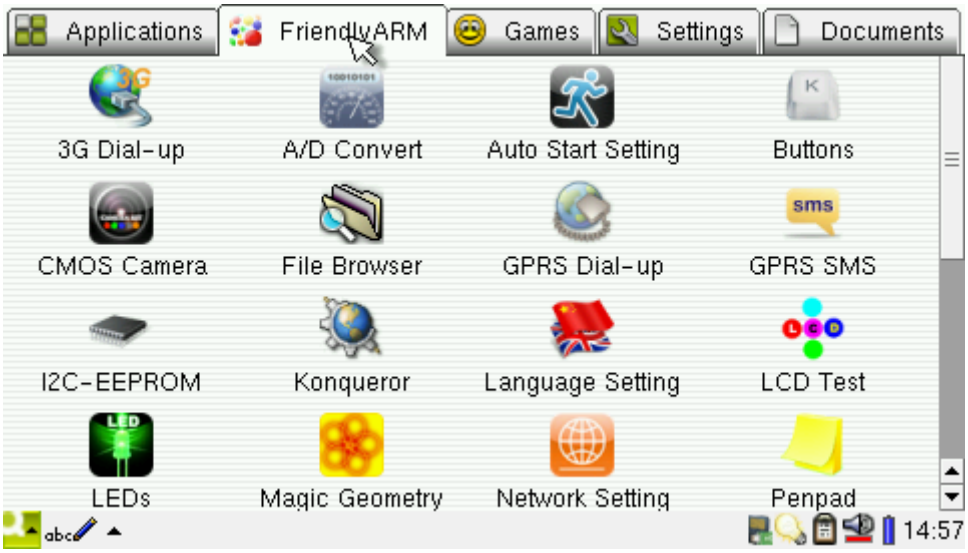
After entering the Qtopia system you will see the following screenshot:



On top of the interface, you will see five icons, which represent five types of programs/files. Single click on anyone you will enter its sub-interface. All of these interfaces are very similar.

In addition, click on the “start” icon on the left bottom of the screen, you will see five sub-menus too, they are the same as the five ones on the top.

Among those programs, the ones in the “FriendlyARM” sub interface are developed or migrated by FriendlyARM. They are only for testing. All the other programs come with the system.



1.3 SMPlayer

The 6410 system has various functions strong enough to process multi-media files (MFC). It supports hard decoding and playing of MPEG4, H.264/H.263 files. The maximum resolution supported is 720x480 30fps or 720x576 25fps. The Mini6410 system also integrates a Post Processor such that it could smoothly and elegantly zoom in and out when playing. This feature achieves extremely good effects when playing in full screen.

MPlayer is an open source media player relying on various open source libraries which enable it to play varied video files and support video devices such as X11, Framebuffer, SDL, DFB. The version used in our system is based on Framebuffer.

MPlayer by itself doesn't have a GUI. There are many available GUIs such as **SMPlayer**, **KMPlayer** and **KPlayer**. We integrated **SMPlayer** into the Mini6410, which is based on Qt4.x libraries (we used the latest QtE-4.7.0) and upgrade it to a media player GUI. For more details please visit the following websites:

Mplayer's official website: http://www.mplayerhq.hu

SMPlayer's official website: http://smplayer.sourceforge.net/

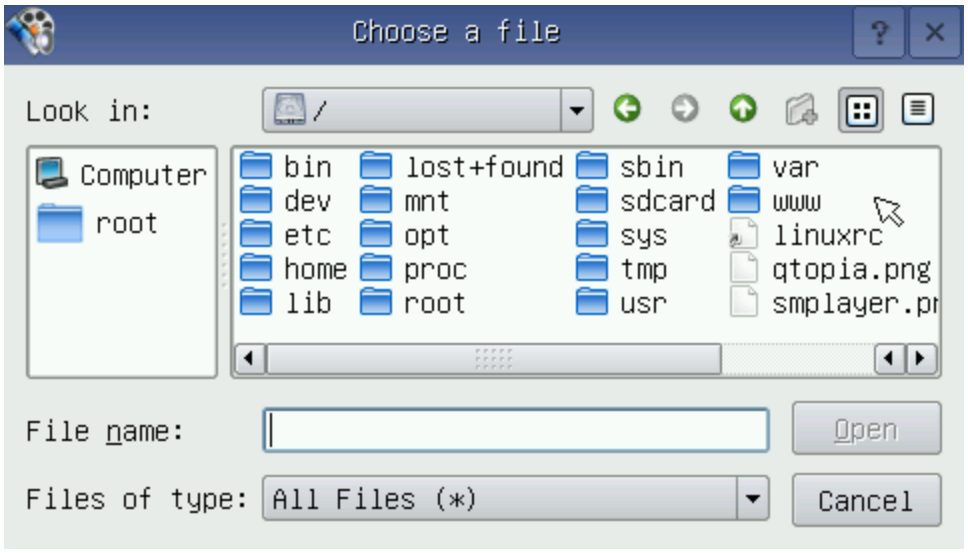
The Linux kernel in our system has included a multi-media driver developed by Samsung. In order to make full use of the 6410 multi-media features we integrated MFC's application libraries into MPlayer. MPlayer in conjunction with SMPlayer is a very strong Linux media player. It can play both MPEG4 and H.264/H.263 files in 4.3"LCD, 7"LCD or monitors with higher resolutions elegantly.

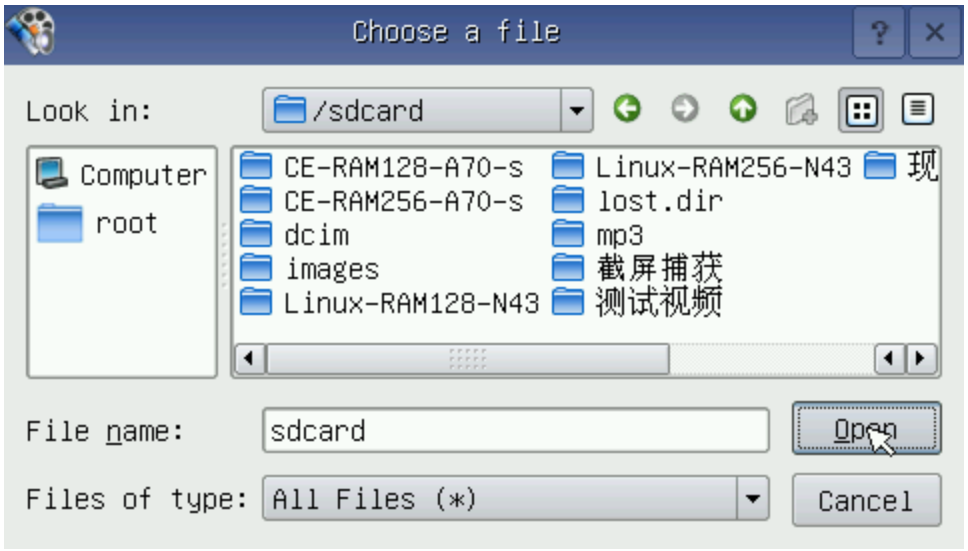
In our shipped CDs there are several test video files for testing.

1.3.1 Play Video with SMPlayer

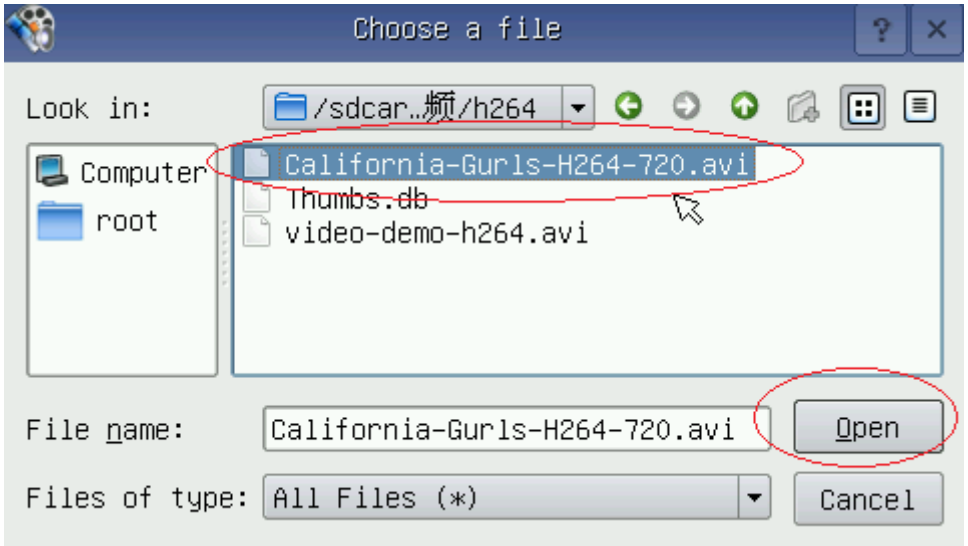
In the "FriendlyARM" tab, click on "SMPlayer"

Click on “Open” or  to select a file you want to play (we use one in “sdcard”)





Locate a file and double click on it to select or click on it and "open"



Now you can enjoy your video:

When it is playing you can click on the screen to pause it and return to the main menu

In the main menu you can adjust the volume, speed, or zoom in/out. Click on the icon  on the upper right or go to “Open-Quit” to quit the application and return to Qtopia2

Note: in the first 5 seconds, a logo “FriendlyARM” will be displayed on the upper left of the screen, suggesting it is developed by us. If you want to customize your player please contact

us.

1.3.2 Convert Video Files

Some video files cannot be directly played by the system, please follow the steps below to convert them before play

Convert to MP4:

With Aimersoft iPhone Converter Suite users can convert a video file to an MP4 that can be played in the Mini6410. It was originally for iPhone, however iPhone1's CPU is 6410 therefore it can be used here as well. Our version is Aimersoft iPhone Converter Suite 1.1.32. We recommend this version and don't guarantee other iPhone versons can work here.

Below is the Aimersoft iPhone Converter Suite's main menu:

Click on iPhone Video Converter you will see the dialog below:

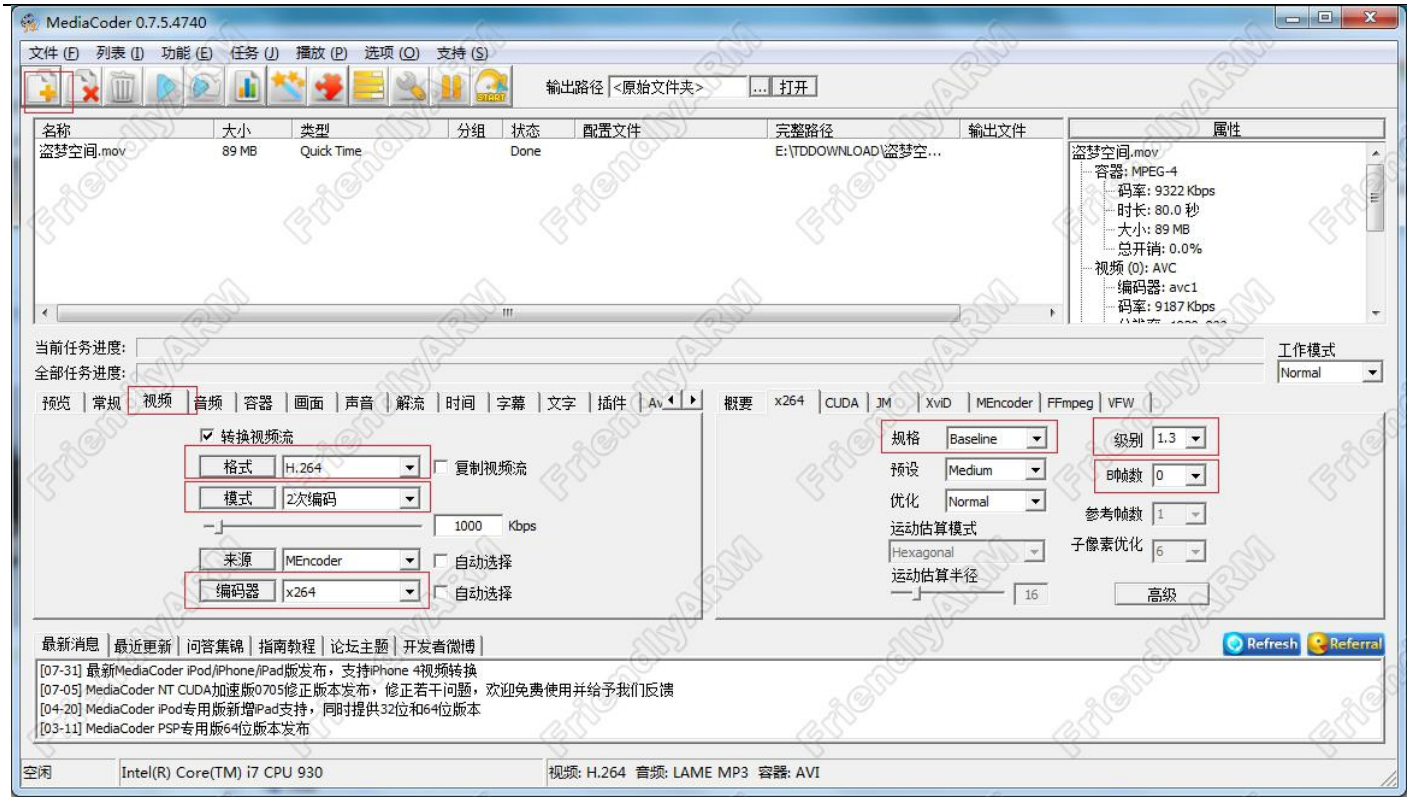
Click on “Open File” to select your file, select “Apple TV MPEG-4 720X432(*.mp4)” and click on “Setup” to introduce the dialog below and follow the settings marked in red

Click on “OK” to return to the main menu and click on “start” to begin conversion

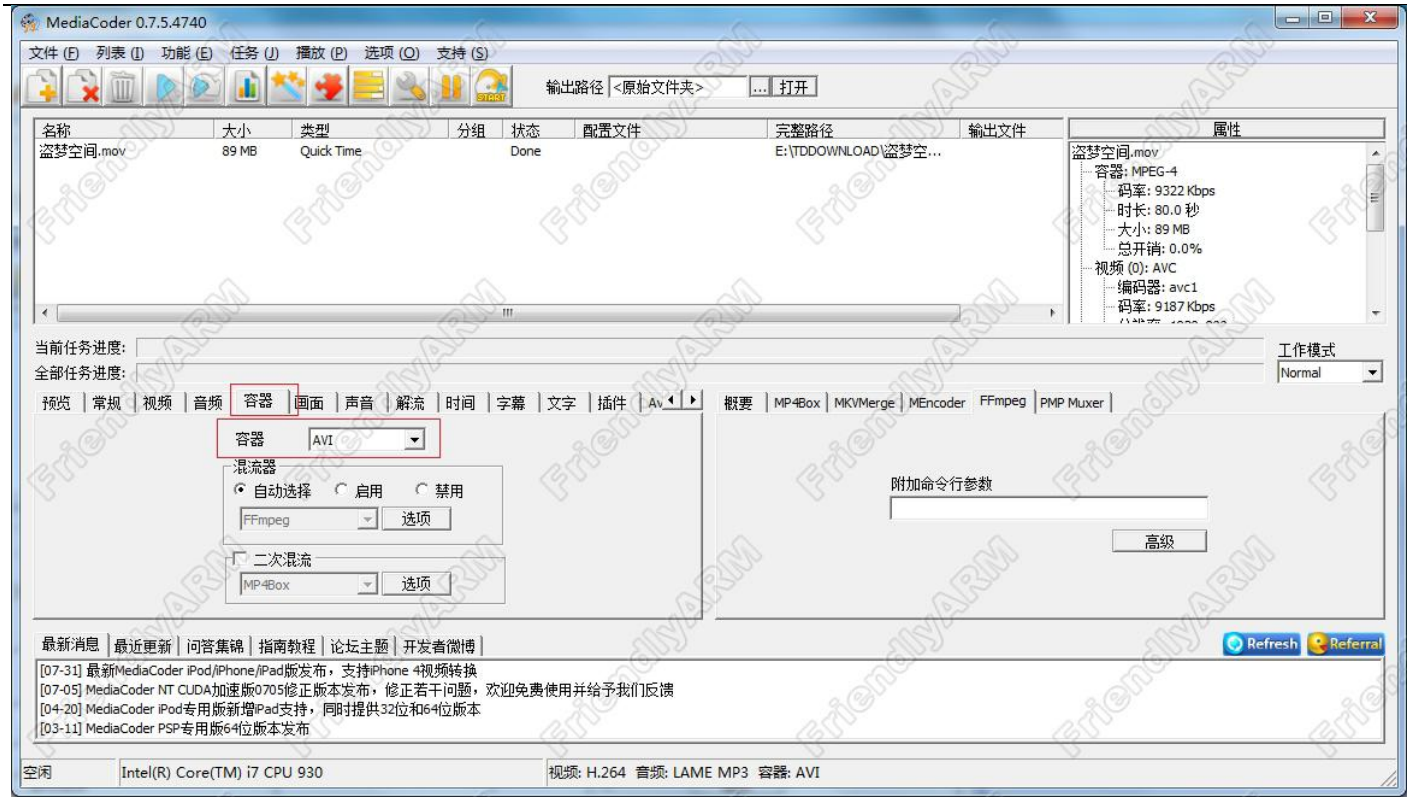
Convert to H264:

With MediaCoder users can play H264 video files. We used MediaCoder 0.7.5.4740 in our system at this time being, which is free. For more details please visit http://www.mediacoder.cn/.

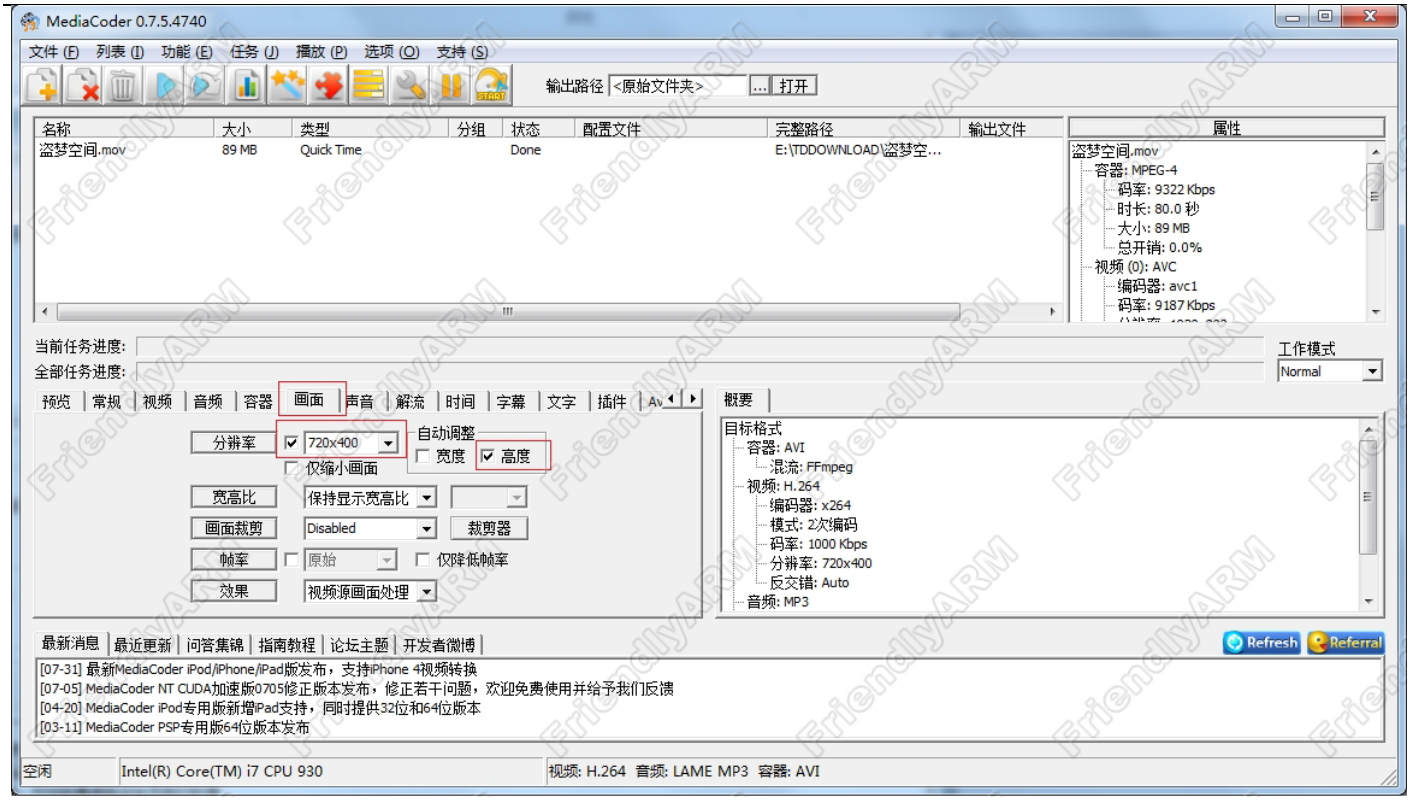
Start MediaCoder and you will see its main menu as below. Click on “+” on the upper left cornor and follow the settings marked in red to configure



Our settings here are based on 6410's decoding capability. Please go forward to set up more items



We set the maximum resolution to 720x400. The maximum resolution 6410 can support is 720x480 30fps. We cannot set it to a higher level.



After setting is done, click on “Start” to begin conversion.

1.4 Play MP3

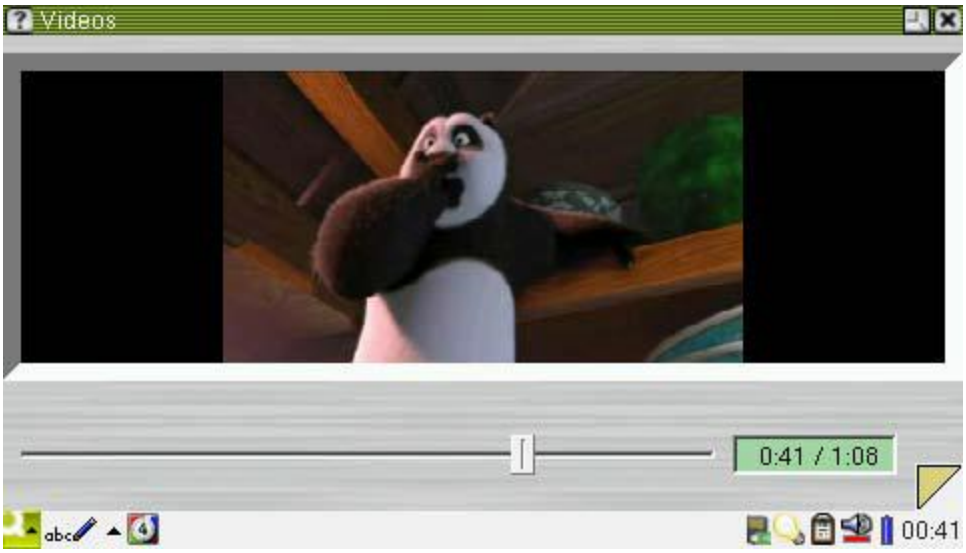
Go to the “Applications” tab and click on “Music” to start a player. In the “Audio” list you can select an MP3 file and “play”.

Note: files listed in “Audio” can all be viewed in the “Documents” tab. You can go to the “Documents” tab and play it there.

1.5 Play Video

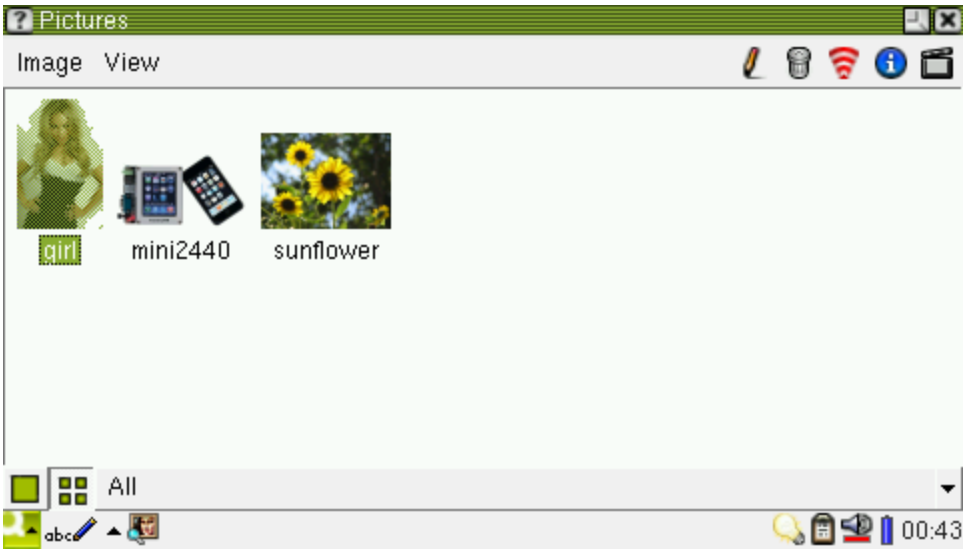
Go to the “Applications” tab and click on “Video”. In the “Video” list you can select a video file and play. This player is integrated in Qtopia it can only do soft decoding thus couldn’t play H.264/H.263/Mpeg4.

Note: files listed in “Video” can all be viewed in the “Documents” tab. You can go to the “Documents” tab and play it there.



1.6 View Pictures

Go to the “Applications” tab and click on “Pictures”. You will see the icons of these pictures in “Documents”.If you have a SD card or flash drive mounted pictures in it will be listed too.



Qtopia 2.2.0 has an image viewing utility which is better than the one in Qtopia 1.7.0 and users can use it to edit images.

1.7 Auto-Mount SD Card/Flash Drive

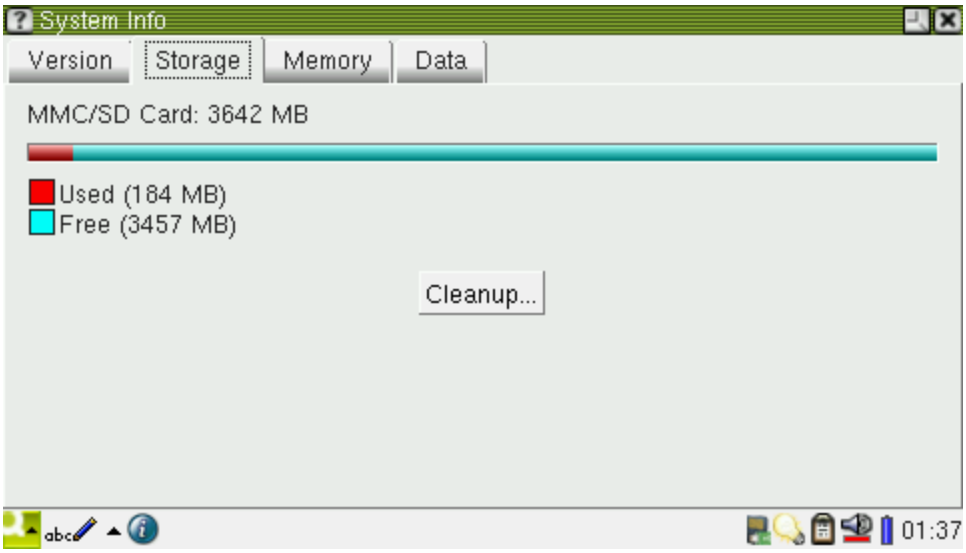
Insert a common or high speed SD card (max memory 32G) or a USB flash stick, moments later a small icon will pop up on the lower right of the screen. The Mini6410 supports simultaneous mounting of the two. Click on the icon you will see the screenshot below, you can remove it safely like what you can do in Windows

All files in the MMC/SD card or the flash drive can be viewed in the “Documents” tab. Their directories will not be displayed.

Auto mounting of a MMC/SD card or a flash drive is developed by FriendlyARM based on a Qtopia 2.2.0 plugin. Now this function can only recognize the first section and can only recognize VFAT/FAT32/FAT16.

Files of other formats may not be recognized correctly.

Click on “Applications -> System Info -> Storage” you will view your SD card or flash drive’s detailed memory information:



1.8 Calculator

Go to the “Applications” tab and click on “Calculator”. You can configure your calculator

to different types by selecting “Simple”, “Fraction”, “Scientific” and “Conversion”.

1.9 Terminal

A terminal is a widely used interactive interface in Linux. Users type commands in a terminal to operate the system. You can set up or open a terminal in various ways:

On system startup, if a terminal is bound to the serial port all its outputs and inputs are to and from this serial port. This is a common Linux way.

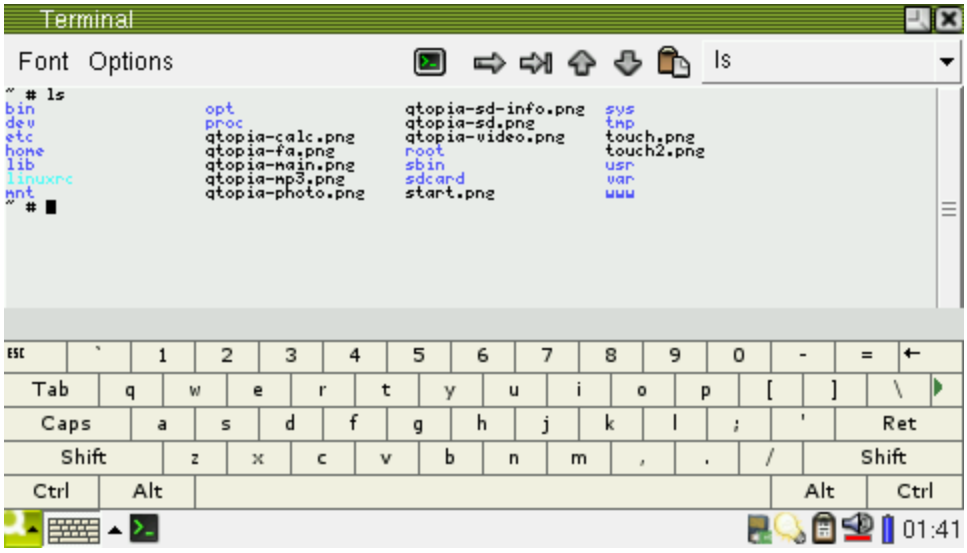
On system startup if a terminal is bound to a graphic device (such as LCD or CRT) and the keyboard is set to the input device then an independent input/output system will be established.

When a graphic device is connected and a GUI is integrated a GUI based “command terminal window” will be established. Users can interact with the system either via a

keyboard or a “soft keyboard”. The latter is what we will talk about.

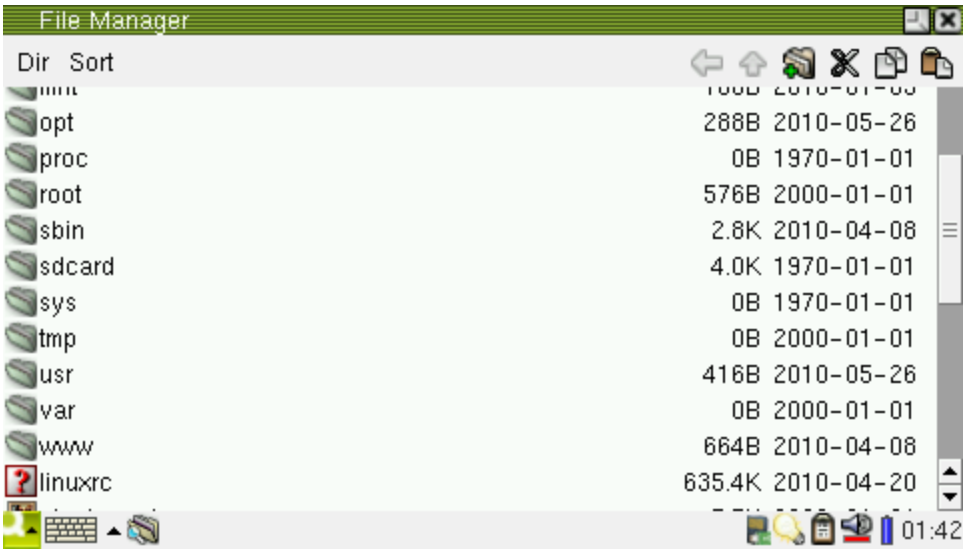
Go to the “Applications” tab and click on “Terminal”. You can either connect a USB keyboard (connect the USB keyboard after system is up) or type on the “soft keyboard” to input your command. You can customize it by clicking on the “Option” menu to set up more configurations.

.



1.10 Manage Files

Go to the “FriendlyARM” tab and click on “File Manager”.



You can browse and manage files and directories

Note: Qtopia-2.2.0 doesn't have this manager, we migrated the one from Qtopia-1.7.0. They have identical functions and interfaces.

1.11 Set up Network

Go to the "FriendlyARM" tab and click on "Network Setting":

From this interface we can set various network parameters:

- Static IP address, default setting is 192.168.1.230
- Mask, default setting is 255.255.255.0
- Gateway, default setting is 192.168.1.1
- DNS, default setting is 192.168.1.1
- MAC address, default setting is 08:90:90:90:90:90

Click on "Save" to save these parameters and they are effective right now. After rebooting the system, these settings will still be there. The configuration file that contains the settings is "/etc/eth0-setting".

Note: the "/etc/eth0-setting" file will not exist after reinstalling the system. Clicking on the "Save" button will generate one. Because all products are tested extensively by us, this file exists in your system. Executing the "ifconfig" command will not change this file. In fact, Qtopia has a network setting utility by itself. But its interface is too complicated and may not

work sometimes. We didn't make any change to this utility however created another one shown above.

1.12 Set up WiFi

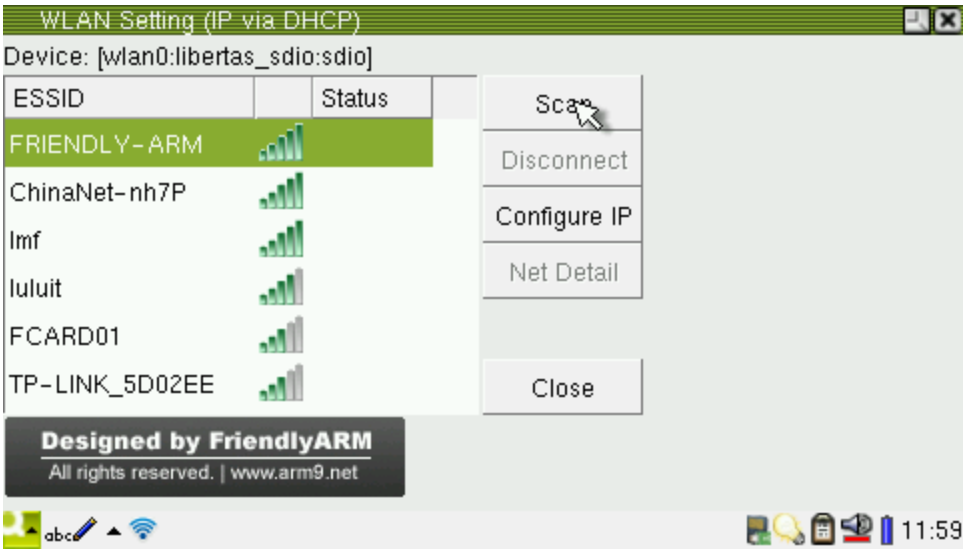
This section will guide to through the steps to set up WiFi. The Mini6410/TINY6410 supports most of the popular USB WiFi cards:

1.12.1 Start WiFi Utility

Go to the “FriendlyARM” tab and click on “WLAN Setting”

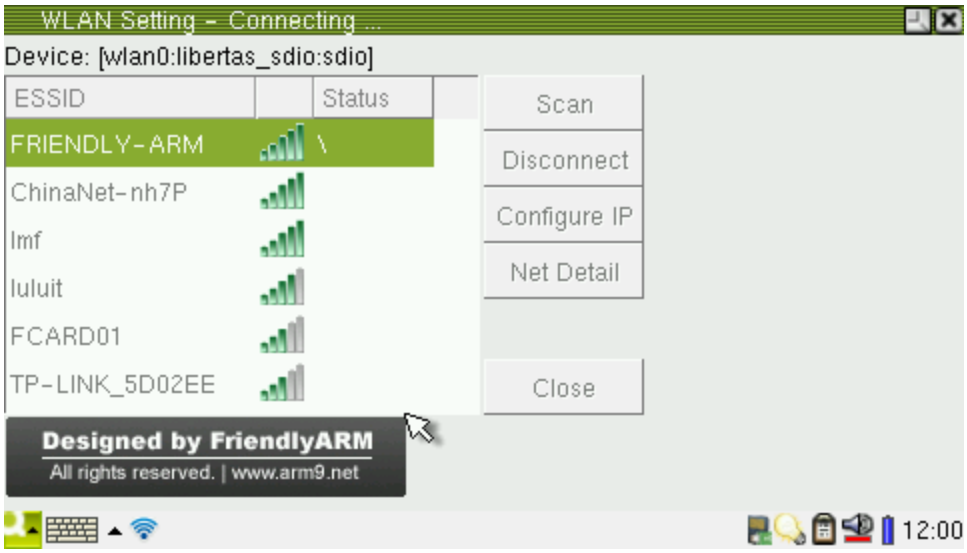
1.12.2 Search for and Connect to Wireless AP

After the WLAN Setting utility is started it will automatically search for wireless AP and list their SSIDs and signal strength. If your utility doesn’t show sources please click on “Scan”

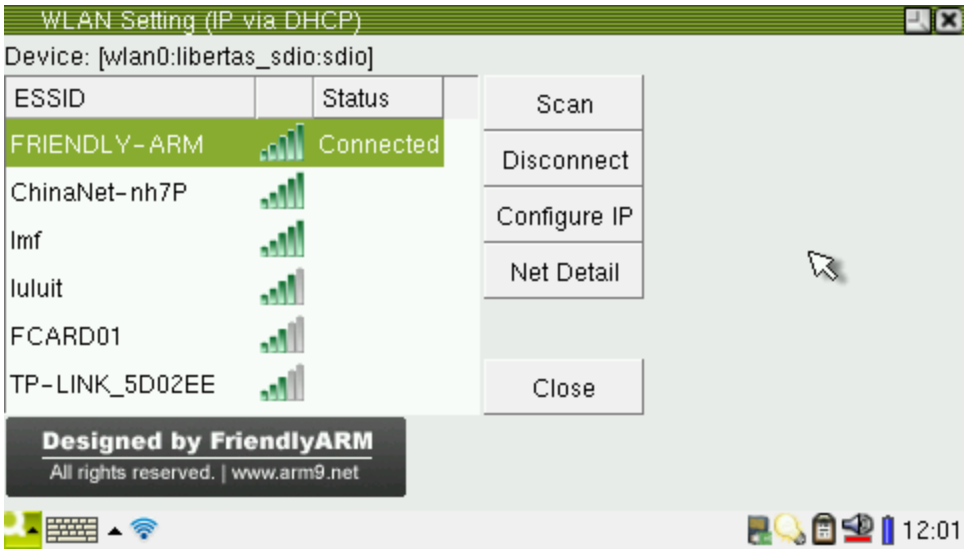


After an AP is found click on its SSID to connect. The following dialog will pop up and you need to input its password:

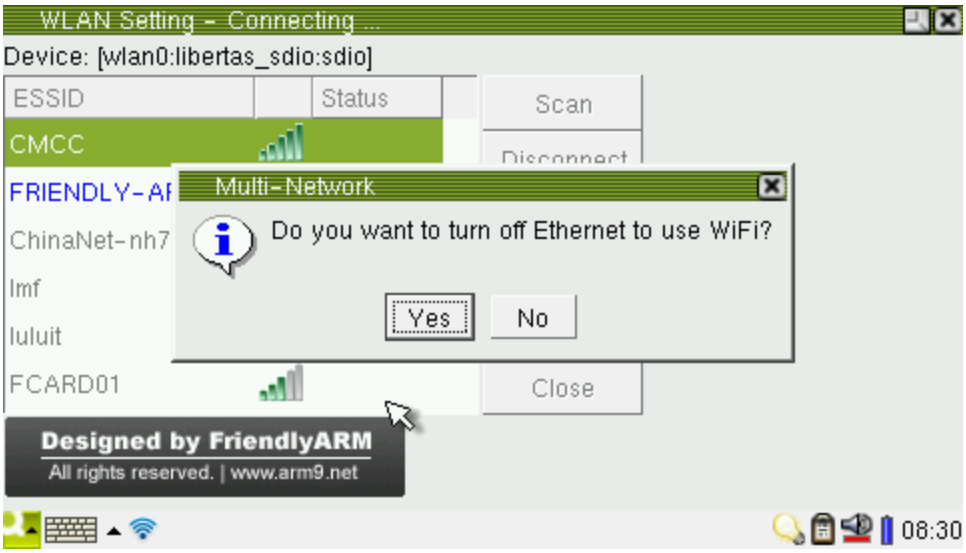
After typing the password (leaving it as blank if no password is set) click on “Connect”



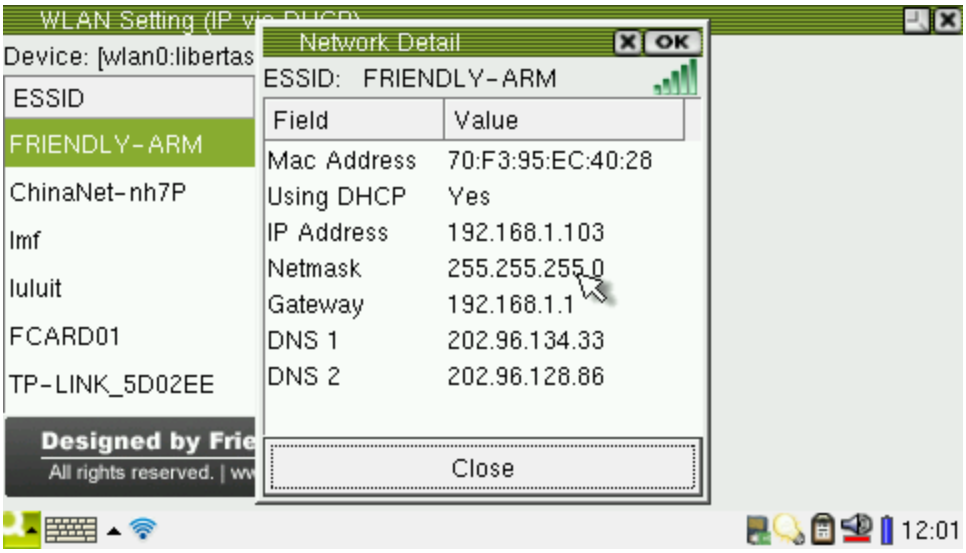
If the connection is successful, its status will show "connected"



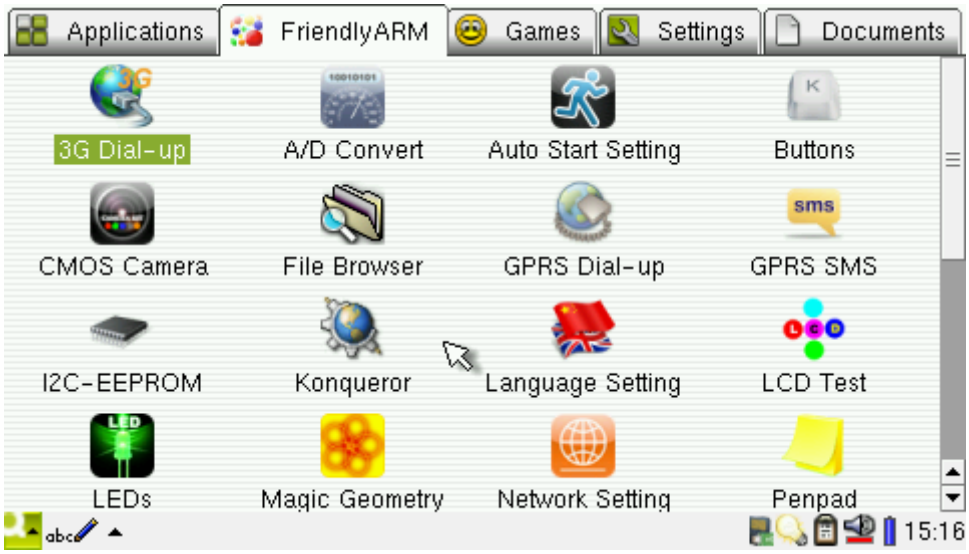
If your Ethernet is connected the following dialog may pop up asking you to disconnect it (ifconfig eth0 down) otherwise some network utilities would connect the Ethernet rather than the WiFi. Clock on "Yes" to close it. To reconnect it you can either start "Network Setting" or type "ifconfig eth0 up".



Click on “Net Detail” to view more detailed network information such as IP, DNS and so on.



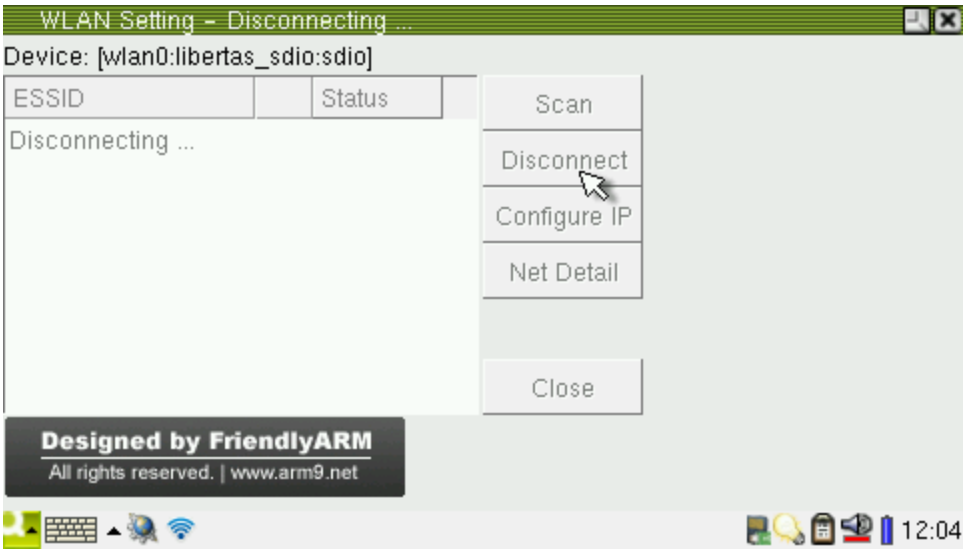
If the WiFi connection is a success, you can minimize its GUI to a small icon by clicking on the “Close” button. To restore its GUI you can click on the small icon



Now you can try your WLAN

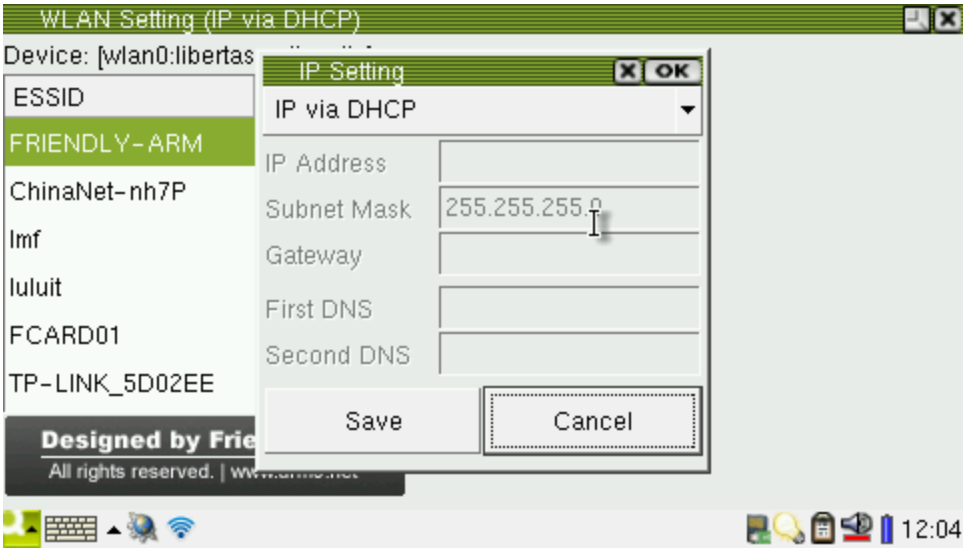
1.12.3 Disconnect WiFi

In the main menu click on “Disconnect” to disconnect the WiFi connection

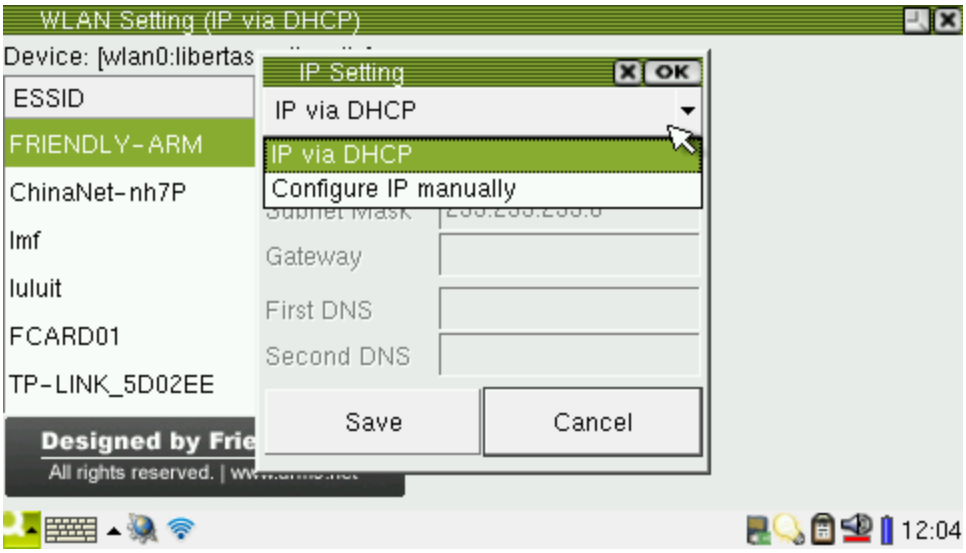


1.12.4 Configure Static IP Address

In the main menu, click on "Configure IP" to start the IP configuration interface:



Click on the pull down menu you will see two options: "IP via DHCP" and "Configure IP manually"



After configuration is done, click on "Save" to keep your settings

1.13 Ping Test

Connect your board to a network, set up your network and you will be able to test PING. Go to the "FriendlyARM" tab and click on "Ping Testing".

If your DNS is set properly you can type either website names or IP addresses. Ping by default sends 4 testing requests. But if you check “ping forever” it will send requests forever.

Note: to ping a website you need to set your gateway and DNS properly and make sure your network in connected to the internet.

Click on “Start” to ping and “Stop” to stop it. To shutdown the ping utility you need to stop it first.

Note: PING is a common network testing utility. All Linux versions and MS Windows have this utility. The PING utility actually calls the ping command and output its result in the GUI.

1.14 Web Browser

Go to the “FriendlyARM” tab and click on “Browser”. Click to start the “soft keyboard” on the lower right of the screen, type a website and click on “Ret” on the keyboard you will be able to visit your website.

Note: the browser utilized in the Mini6410 is Konqueror/Embedded, which is open source

1.15 LED Test

Go to the “FriendlyARM” tab and click on “LED Testing”.

You will see that only “Stop led-player” is enabled because the system started the led-player service. When booting the system you will see LEDs round robin flashing which

is manipulated by this service. To control a single LED you need to stop this service to release the LED resources.

Click the “Stop led-player” button it will change to “Start led-player”, all LEDs on the board will be turned off and all the buttons in the “LED Control” framework will be enabled

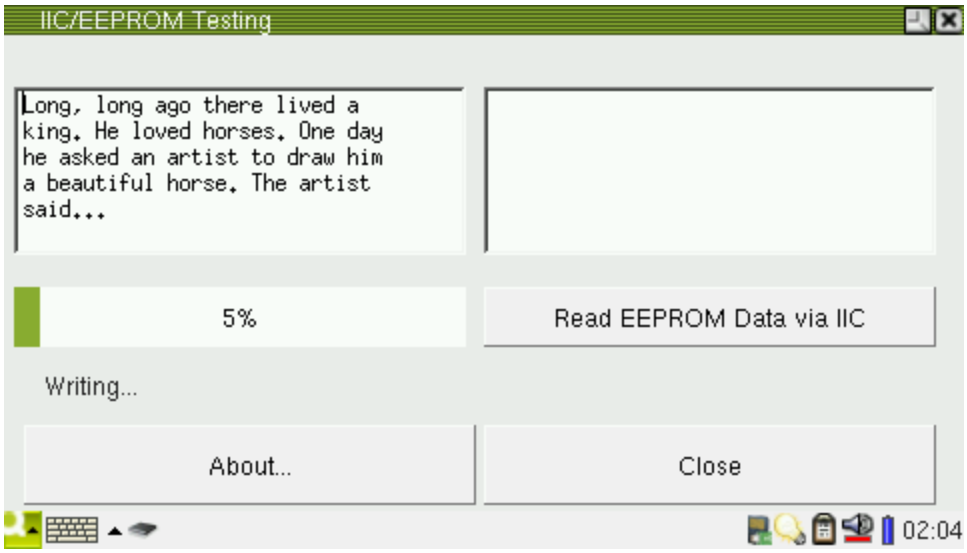
Click on “All On” you will turn on all LEDs. Click on “All Off” you will turn off all LEDs. Checking any box on the left will turn on its corresponding LED and unchecking the box will turn the LED off.

When you close the LED Testing GUI the led-player service will be restarted.

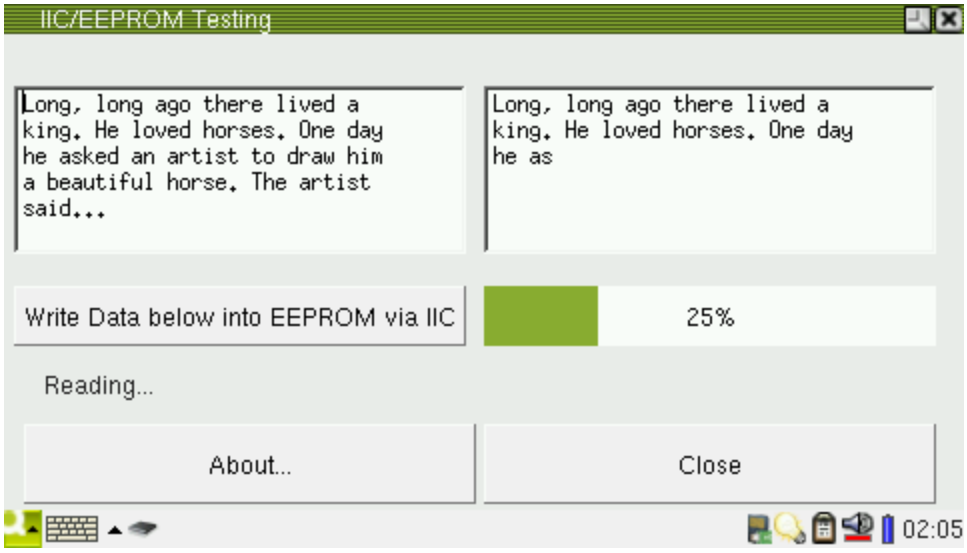
1.16 EEPROM Test

Go to the “FriendlyARM” tab and click on the “I2C-EEPROM” icon to open the interface. Open the “soft keyboard” on the task bar, write some characters in the write area, click on the

"Write Data below into EEPROM via IIC" button, the button will change to a process bar indicating the writing process.

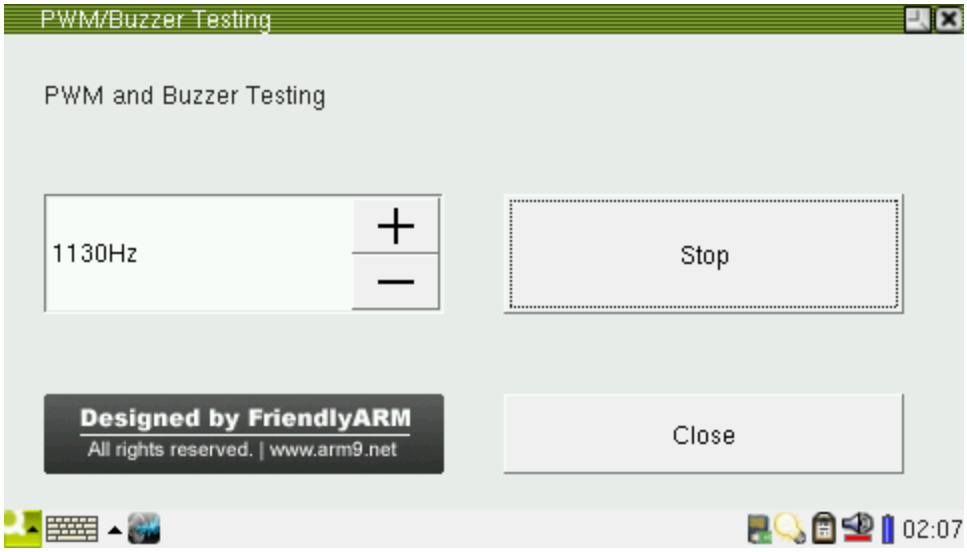


Click on the "Read Data via IIC" button, it will change to a process bar too indicating the reading process



1.17 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



1.18 Serial Port Assistant

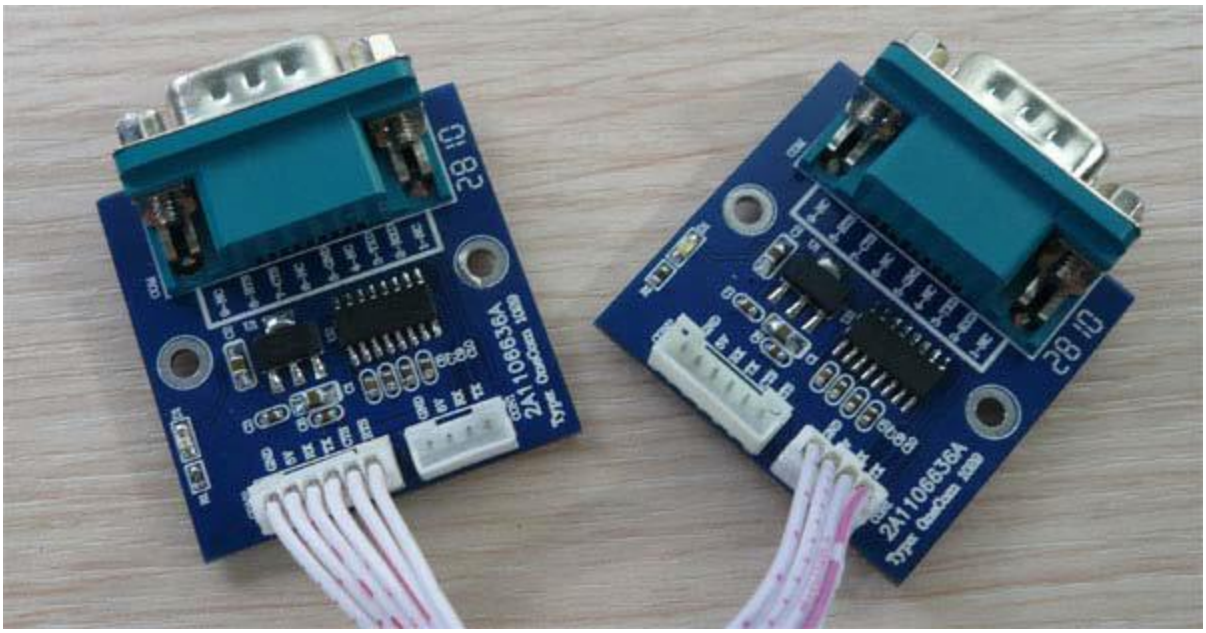
Note: before start this program please connect the serial port your want to test to your board.

- 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

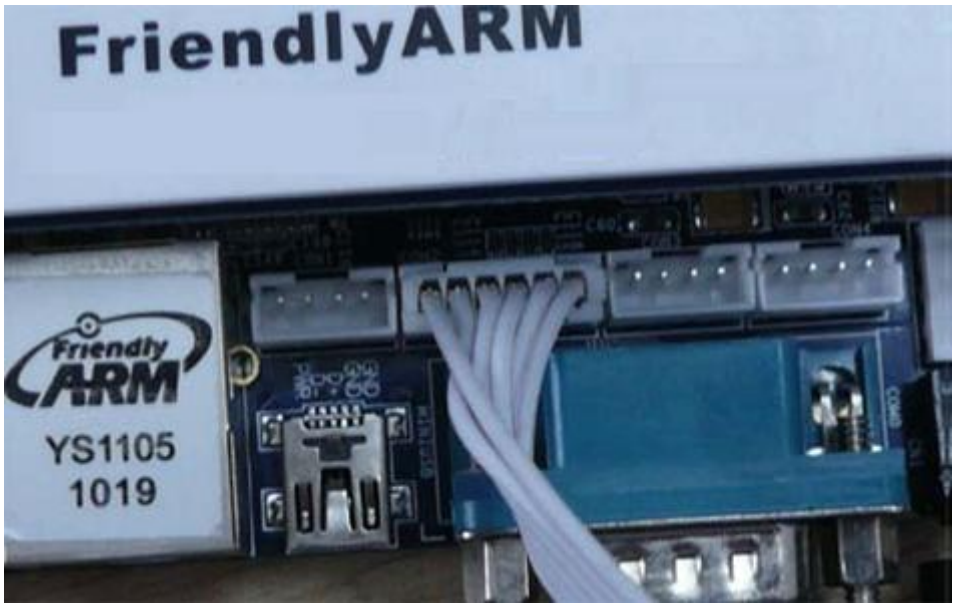
"OneCom" RS232 conversion module) When connect the ports to a PC, please make sure to use a correct serial cable (cross serial line or direct serial line).

- 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 us

e a USB hub to extend your serial ports.



Connect your serial port extension board to the Mini6410'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

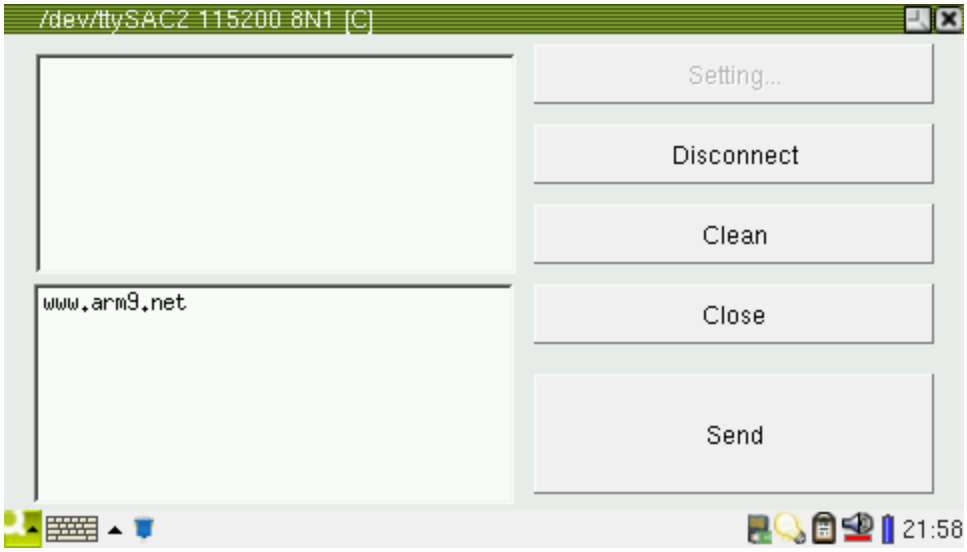
- Flow Control: None

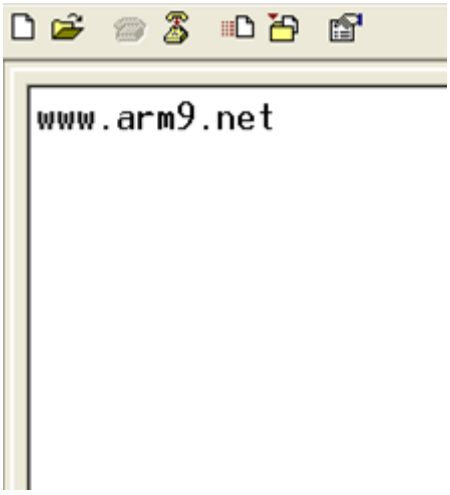
- Stop Bits: 1

- [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:

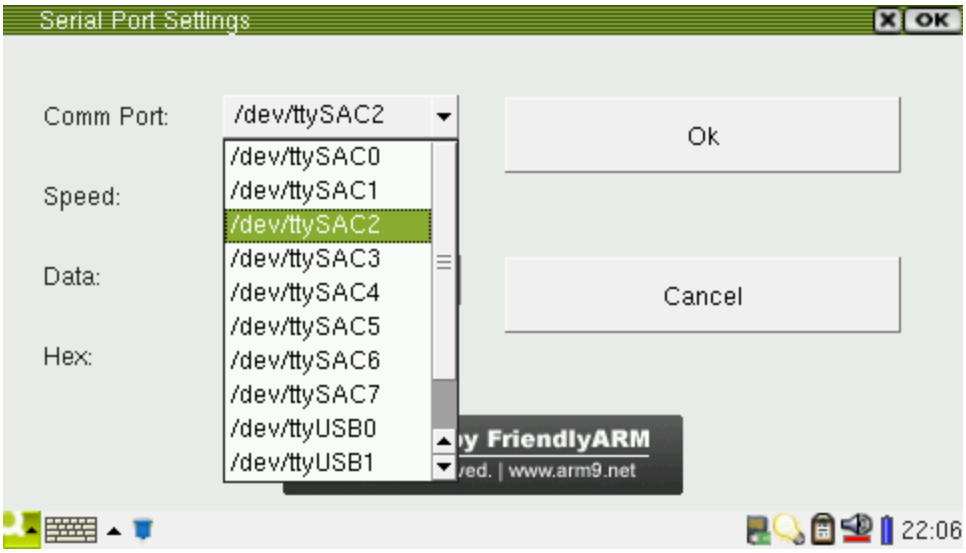
Comm Port: you can choose “/dev/ttySAC0,1,2” or the USB to Serial “/dev/ttyUSB0,1,2,3”.

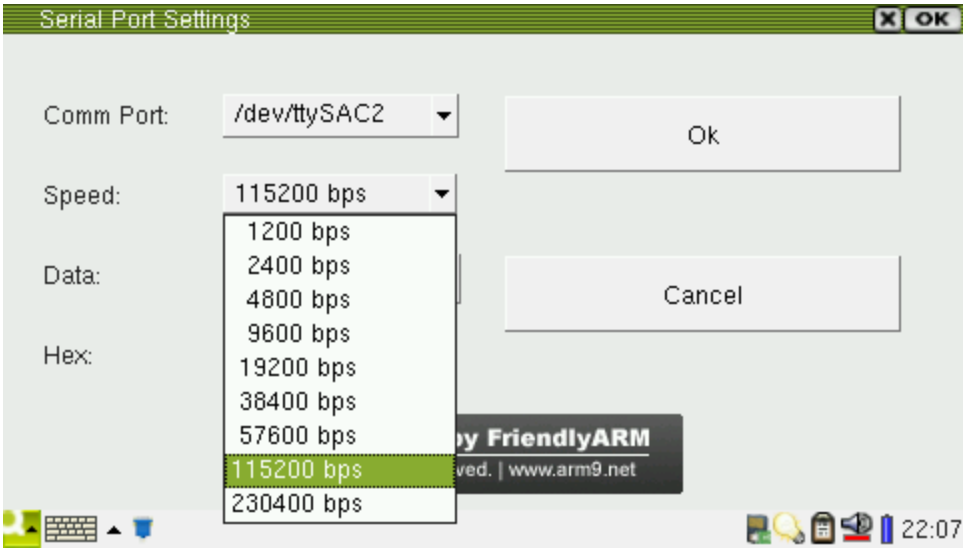
Note: in this utility, SAC0 corresponds to CON1, SAC1 corresponds to CON2 and etc.

Speed: bits per second

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

Hex: input and output data in Hex format





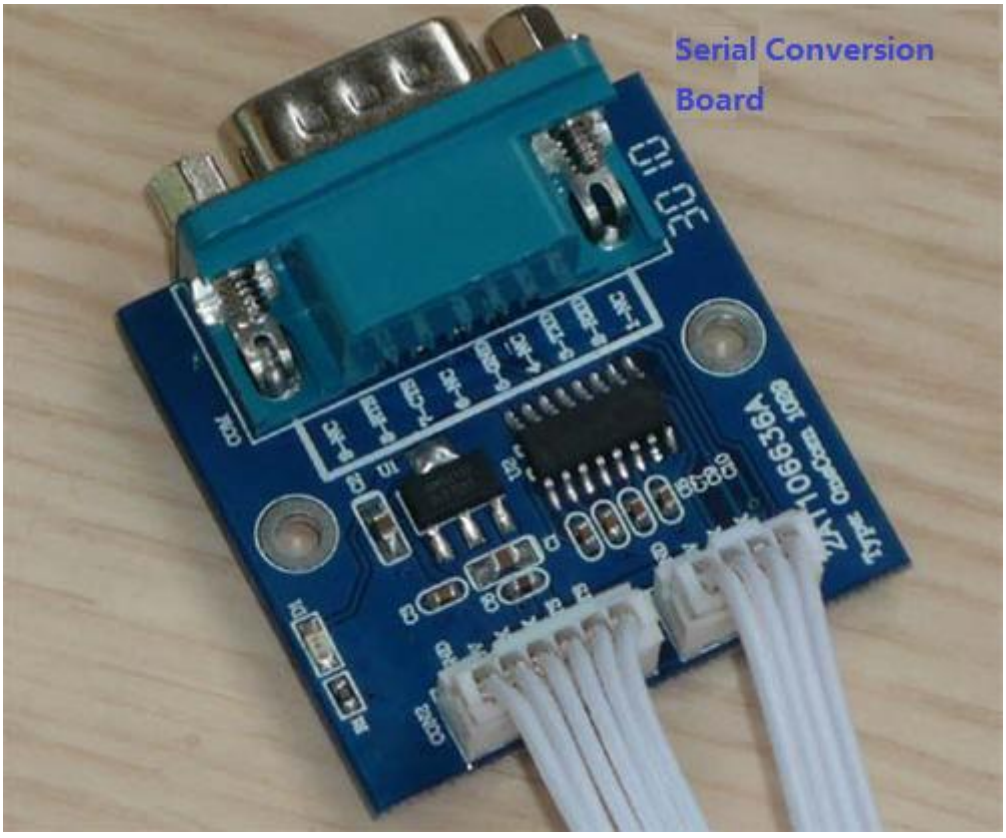
1.19 Connect to Internet via GPRS Modem

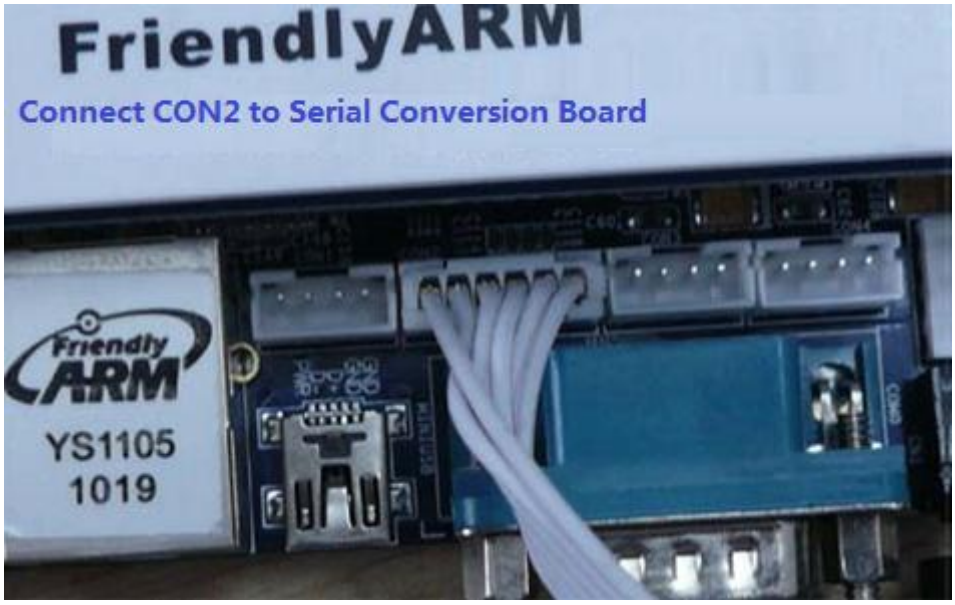
You can connect to the internet via common a GPRS modem. Our shipped package includes a GM2403 modem which incorporates Wavecom's industrial Q2403A module and supports GSM/GPRS 900M/1800M. For more details please refer to its manual.

You can connect to a Modem via either a serial cable or a USB cable.

(1) Connect via Serial Cable

To Connect a GPRS modem via a serial cable you need a four-wire serial cable (also called five-wire cable and the fifth is grounded): TXD，RXD，CTS and RTS. The Mini6410's first serial port is four-wired however it is reserved for the console terminal; the second serial port is four-wired too which corresponds CON2 (whose device name is /dev/ttySAC1) and is TTL. We need to convert it to RS232 before it can be connected to a Modem. You can use our offered serial conversion board (model:OneCom2) or make your own

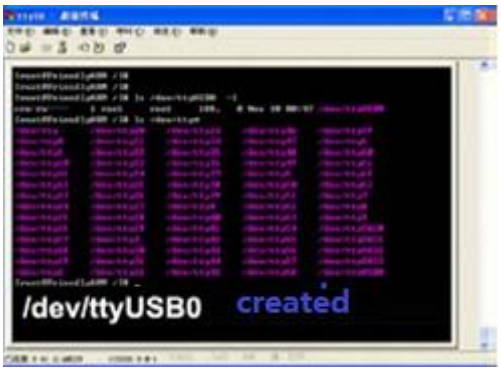




Note: when the dialing utility is operating the serial device it will set CTS and RTS. This operation is necessary. Using a serial port that has CTS/RTS ensures integrity and security of transmitted data.

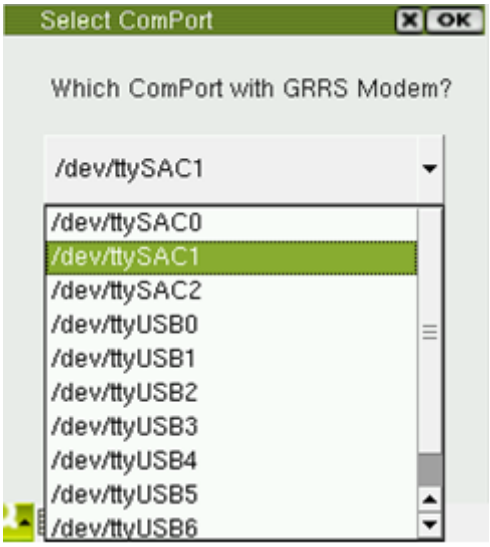
(2) Connect via USB to Serial cable

If you don't have a conversion board mentioned above you can use a USB to Serial connector too. Our kernel supports most of the popular USB to Serial connectors which supports all serial functions including CTS and RTS

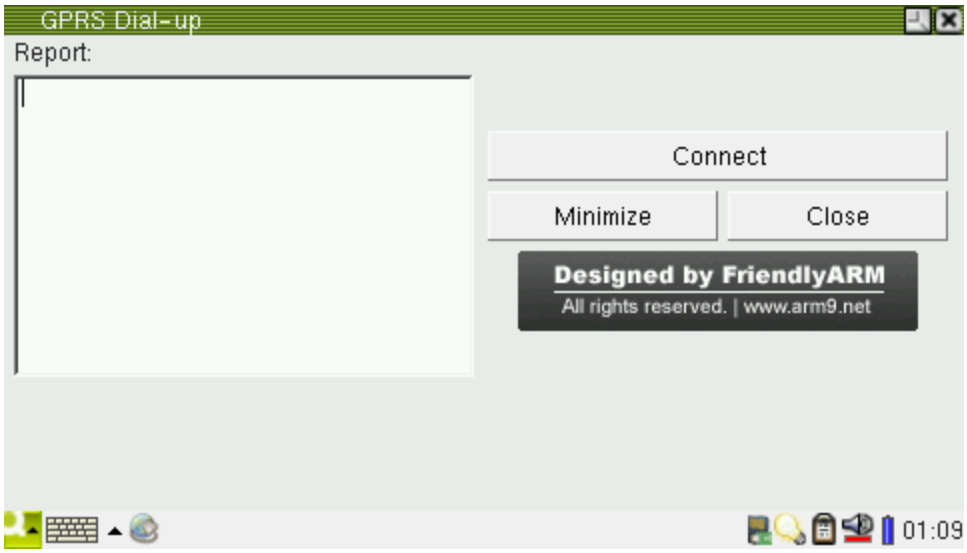


Note: after a USB to Serial device is inserted, you will find a new device "/dev/ttyUSB0" or "/dev/ttyUSB1" listed in the "/dev" directory

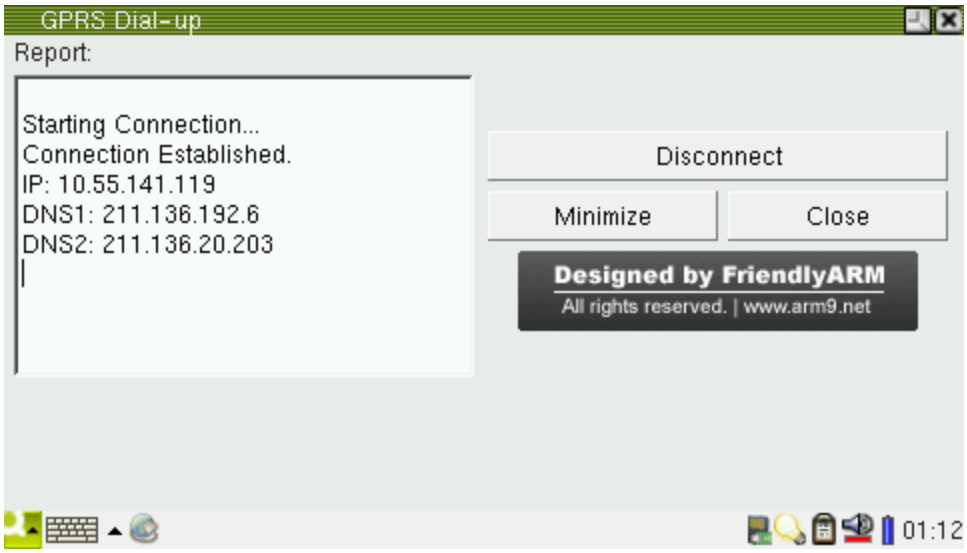
Go to the "FriendlyARM" tab and click on the "GPRS Dialing" icon you will see a configuration dialog pop up. If You are using your CON2 conversion board you need to select "/dev/ttySAC1" and select "/dev/ttyUSB0"if you are using a USB to Serial convertor.



We select /dev/ttySAC1 and click on OK to continue.



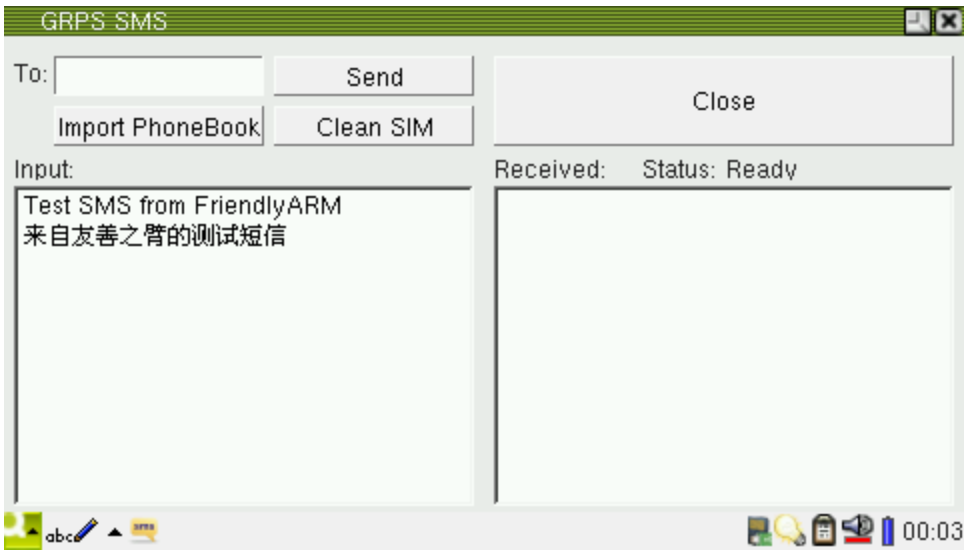
The dial up window is straightforward and you just click on “Connect” to begin dialing. After it is successful it will show an allocated IP and DNS.



Clicking on “Disconnect” closes the connection. Clicking on “Minimize” minimizes the window. We usually minimize it and start a browser to surf the internet.

1.20 Single/Group-Send Messages via GPRS Modem

Please go to “FriendlyARM” and click on “GPRS SMS”



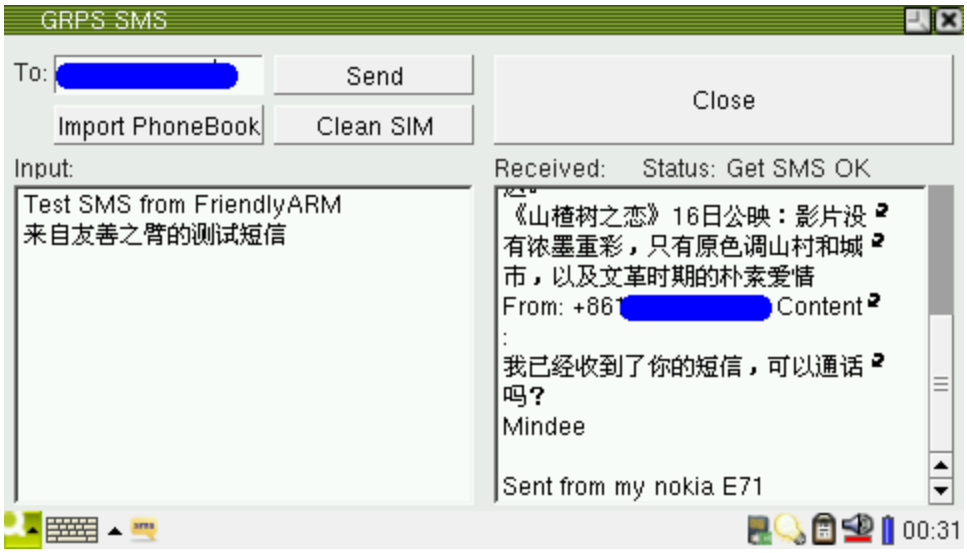
If the connection is a success, the status will show “Ready”. Occasionally it shows “Device Initing…” and this suggests that the previous shutdown was an abnormal operation or the previous Modem connection wasn’t disconnected or your connection isn’t good. We recommend users to use a USB to Serial convertor to connect a Modem.

(1) Single sending messages

Please input your target cell phone number in the edit box on the right side of “To:”, type your message in the edit box below “Input” and click on “Send”.

If the sending is a success and the telecommunication service provider sends back an acknowledgement you will see the “Status” shows “Get SMS OK”.

You can send messages to your own cell phone for testing too



The format of the cell phone number varies in different countries:

In mainland China it is “+8613800138000” or “13800138000”.

On Abroad you need to add a country code prior to the number e.g. “+4423645789”. The number after “+” composes of the country code first and the cell phone number then.

Note: there is no Chinese input utility in the system therefore if you want to send message in Chinese you have to copy and paste it from elsewhere.

(2) Group sending messages

To group send messages you need to edit a "phonebook.txt" and it should be in the root directory of the SD card. Note: the name of the file cannot be changed. Its contents could be as follows:

Mindee
13800138000
Mike
13800138000
Jason
+8613800138000

Names can be ignored:
13800138000
13800138000
+8613800138000

Insert your SD card, click on "Import Phonebook" to import your phonebook, click on "Send" and your messages will be sent out.

Note: this utility itself doesn't save your received messages. Your messages will be saved in your SIM card. If your card is full, you need to "Clean SIM" to remove your obsolete messages.

1.21 Dial-up via 3G Network Card

There are three popular 3G systems WCDMA, CDMA2000 and TD-SCDMA. They require different 3G Modems. The most popular one is the USB 3G Modem, usually called "USB 3G network card" or "USB network card". Our dial-up utility can detect and drive various USB network cards.

We will take HUAWEI E1750 as an example to show you how to use it.

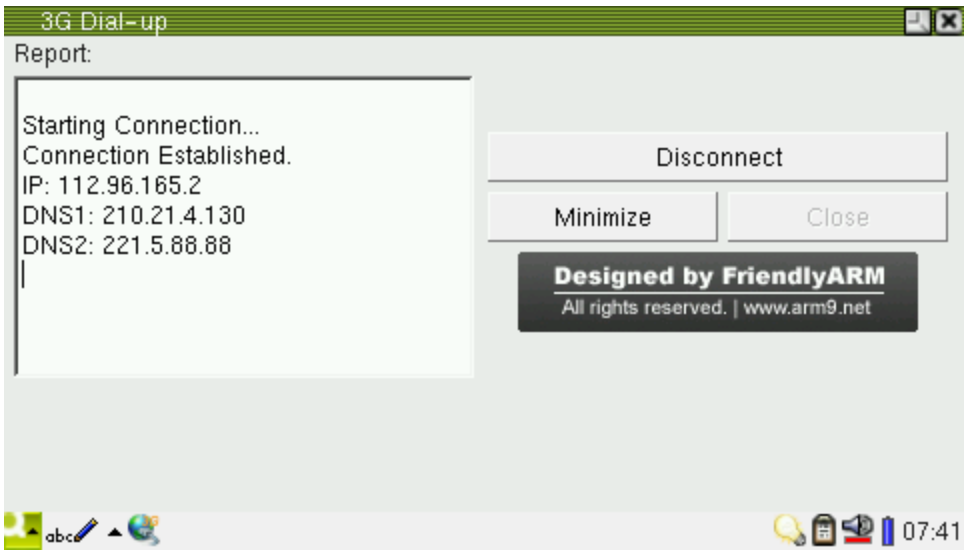
Step1: please get a 3G SIM card ready

Step2：insert the SIM card into the network card

Step3：plug the network card into your board

Step4: power on and start the 3G dial-up utility. It will automatically list all the detected signals. Click on “OK” to continue

Step5: click on “Connect” to start connecting. If the connection is successful it will show the following dialog



Step6: “Minimize” the dial-up utility, open a browser and you will be able to try surfing the internet!

Below is a list of USB 3G network cards that have been verified by FriendlyARM working with our utility:

Huawei E169 (CDMA2000)
Huawei E1750/E1550 (WCDMA)
ZTE AC581 (CDMA2000)
ZTE AC8710 (CDMA2000)
ZTE MU351 (TD-SCDMA)
ZTE 6535-Z
ZTE AC2710 (EVDO)
ZTE AC2726
ZTE K3520-Z
ZTE K3565
ZTE MF110 (Variant)

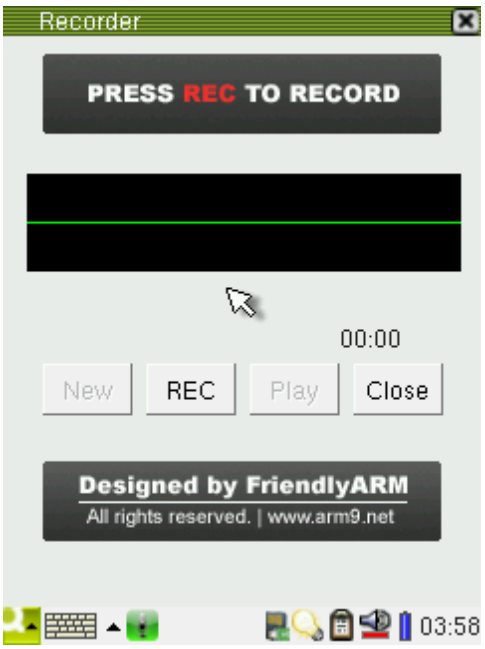
ZTE MF112
ZTE MF620 (aka "Onda MH600HS")
ZTE MF622 (aka "Onda MDC502HS")
ZTE MF628
ZTE MF638 (aka "Onda MDC525UP")
ZTE WCDMA Stick from BNSL
HuaXing E600 (NXP Semiconductors "Dragonfly")
Huawei E1612
Huawei E1690
Huawei E180
Huawei E270+ (HSPA+ modem)
Huawei E630
Huawei EC168C (from Zantel)
Huawei K3765
Huawei K4505
Huawei R201
Huawei U7510 / U7517
Huawei U8110 (Android smartphone)
Onda MW833UP
A-Link 3GU
AT&T USBConnect Quicksilver (made by Option, HSO driver)
AVM Fritz!Wlan USB Stick N
Alcatel One Touch X020 (aka OT-X020, aka MBD-100HU, aka Nuton 3.5G), works with Emobile D11LC
Alcatel X200/X060S
Alcatel X220L, X215S
AnyDATA ADU-500A, ADU-510A, ADU-510L, ADU-520A
Atheros Wireless / Netgear WNDA3200
BSNL Capitel
BandLuxe C120
BandRich BandLuxe C170, BandLuxe C270
Beceem BCSM250
C-motech CGU-628 (aka "Franklin Wireless CGU-628A" aka "4G Systems XS Stick W12")
C-motech CHU-629S
C-motech D-50 (aka "CDU-680")
Cricket A600
EpiValley SEC-7089 (featured by Alegro and Starcomms / iZAP)
Franklin Wireless U210
Hummer DTM5731

InfoCert Business Key (SmartCard/Reader emulation)
Kyocera W06K CDMA modem
LG HDM-2100 (EVDO Rev.A USB modem)
LG L-05A LG LDU-1900D EV-DO (Rev. A)
LG LUU-2100TI (aka AT&T USBConnect Turbo)
Motorola 802.11 bg WLAN (TER/GUSB3-E)
MyWave SW006 Sport Phone/Modem Combination
Nokia CS-10
Nokia CS-15
Novatel MC990D
Novatel U727 USB modem
Novatel U760 USB modem
Novatel Wireless Ovation MC950D HSUPA
ONDA MT505UP (most likely a ZTE model)
Olivetti Olicard 100 and others
Olivetti Olicard 145
Option GlobeSurfer Icon 7.2
Option GlobeSurfer Icon 7.2, new firmware (HSO driver)
Option GlobeTrotter EXPRESS 7.2 (aka "T-Mobile wnw Express II")
Option GlobeTrotter GT MAX 3.6 (aka "T-Mobile Web'n'walk Card Compact II")
Option GlobeTrotter HSUPA Modem (aka "T-Mobile Web'n'walk Card Compact III')
Option iCON 210
Option iCON 225 HSDPA
Philips TalkTalk (NXP Semiconductors "Dragonfly")
Rogers Rocket Stick (a Sony Ericsson device)
Royaltek Q110 - UNCONFIRMED!
ST Mobile Connect HSUPA USB Modem
Sagem F@ST 9520-35-GLR
Samsung GT-B3730
Samsung SGH-Z810 USB (with microSD card)
Samsung U209
Sierra Wireless AirCard 881U (most likely 880U too)
Sierra Wireless Compass 597
Siptune LM-75 ("LinuxModem")
Solomon S3Gm-660
Sony Ericsson MD300
Sony Ericsson MD400
Toshiba G450
UTStarcom UM175 (distributor "Alltel")

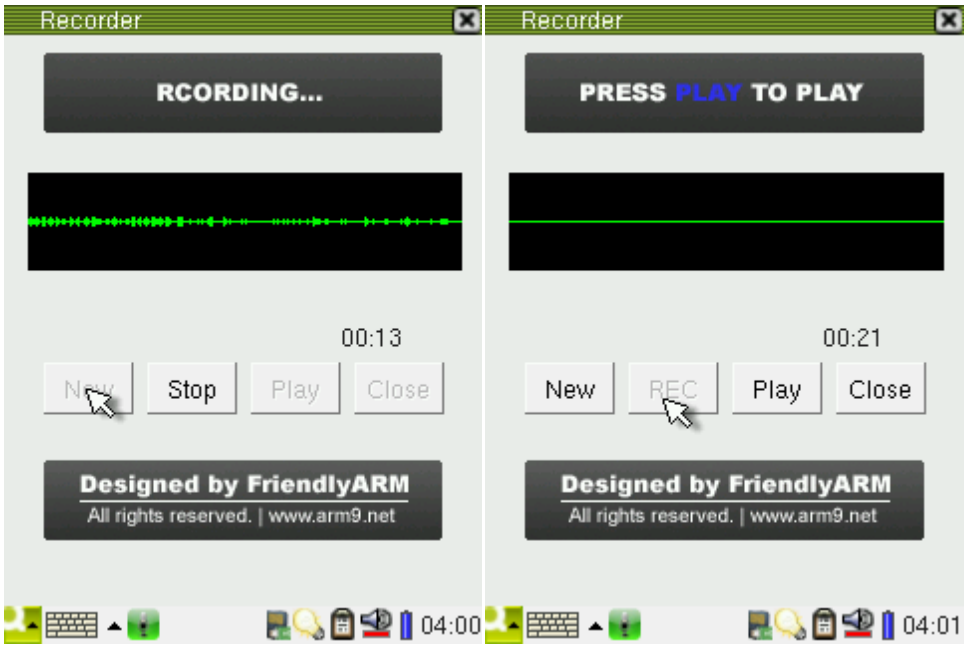
UTStarcom UM185E (distributor "Alltel")
Vertex Wireless 100 Series
Vodafone (Huawei) K4605
Vodafone (ZTE) K3805-Z
Vodafone MD950 (Wisue Technology)
Zydas ZD1211RWWLAN USB, Sphairon HomeLink 1202 (Variant 1)
Zydas ZD1211RW WLAN USB, Sphairon HomeLink 1202 (Variant 2)

1.22 Audio Recording

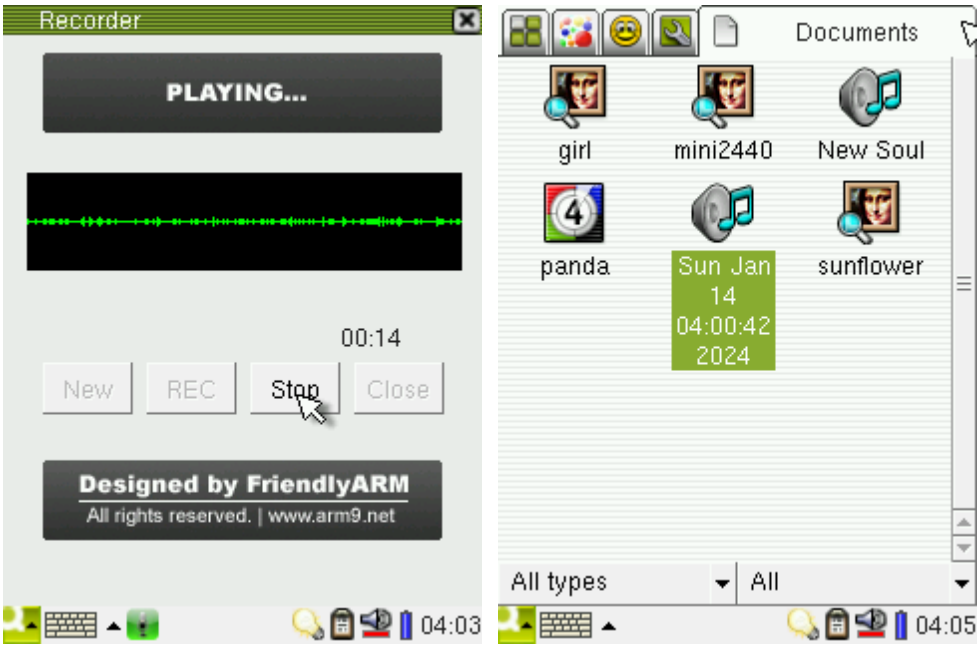
Go to the “FriendlyARM” tab and click on the “recorder” icon:



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.

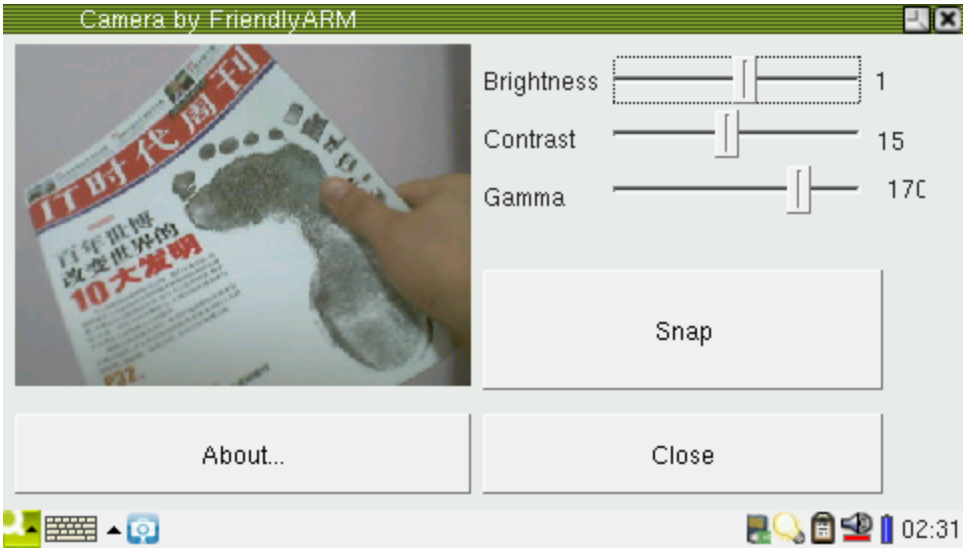


Note: Qtopia 2.2.0 has a recorder utility by itself. But it cannot record audio. We leave it as what it is.

1.23 Work with USB Camera

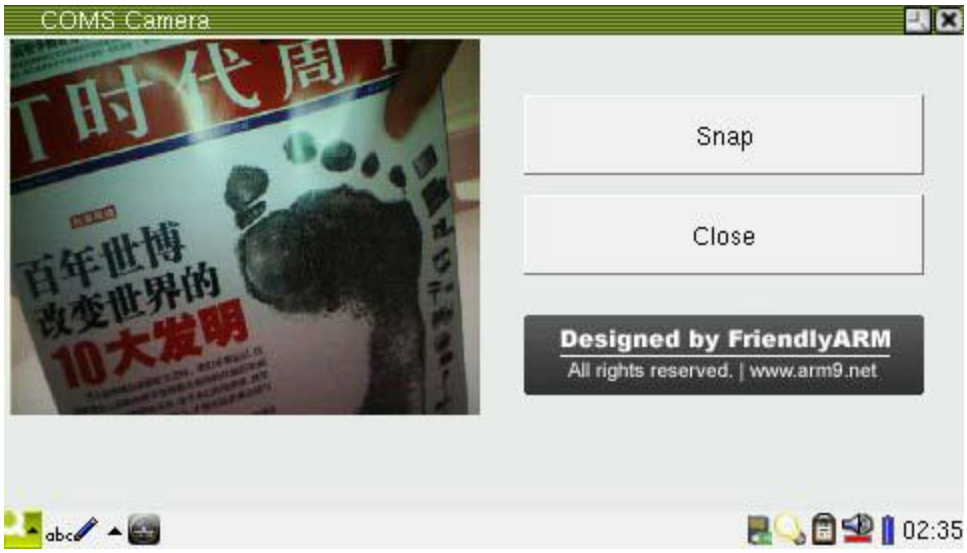
You can use any USB camera with our system which already has drivers for all existing USB cameras. Plug your camera to the USB host port on the board, click on the “USB Camera” in the “FriendlyARM” tab you will see a dynamic preview interface. Click on the “Snap” button you will take a picture which will be saved in the “Documents”. This utility has provides functions to adjust brightness, contrast and gamma value. When you start this utility, it will read the camera preset parameters.

Note: although the system already has drivers for USB cameras, each camera might have different output format. Since we cannot collect all cameras this utility would only work for some common cameras, if your camera doesn’t work with our system please contact us.



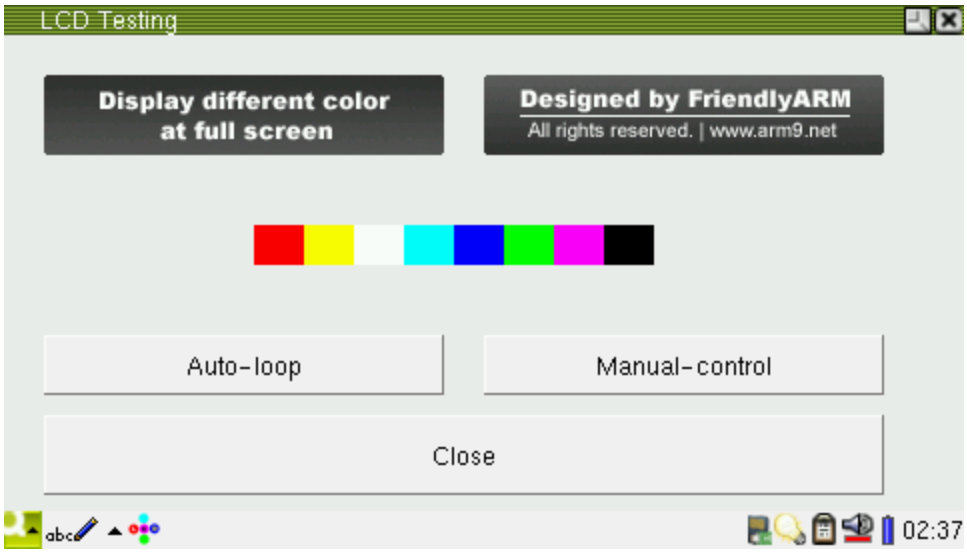
1.24 Preview with Camera

To launch the preview utility you need to use our shipped CMOS CAM130 module which also works with the Mini2440 system. Connect the module to your board, power on, go to the “FriendlyARM” tab and click on “CMOS Camera”. Clicking on “Snap” takes pictures of what your are previewing. After a picture is taken, “Snap” changes to “Continue”. Click on it you will be able to preview again and the picture you just took will be saved in “Documents” (located in “/root/Documents/image/jpeg”). Click on the picture you just took in “Documents” you will see it is opened in Qtopia’s “image” utility.



1.25 LCD Test

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



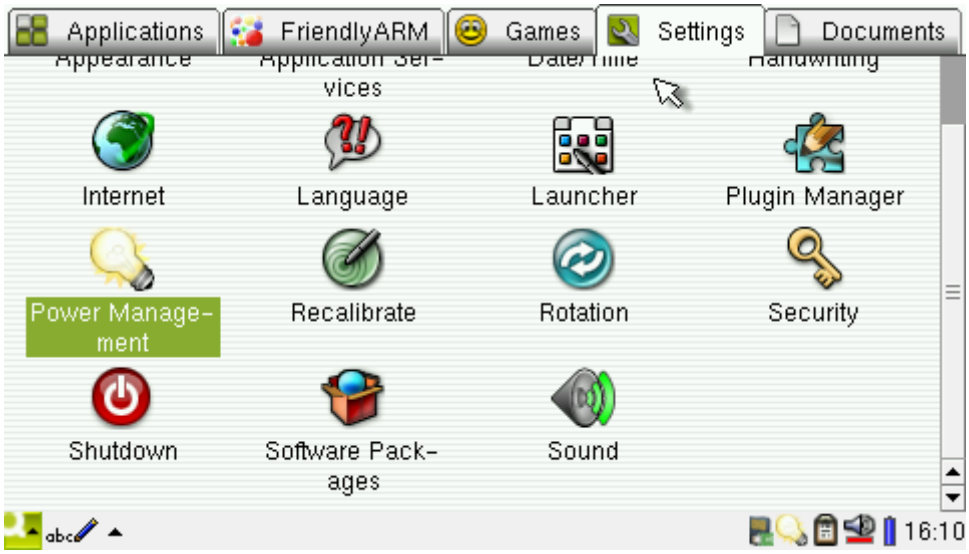
This utility has two modes: auto and manual

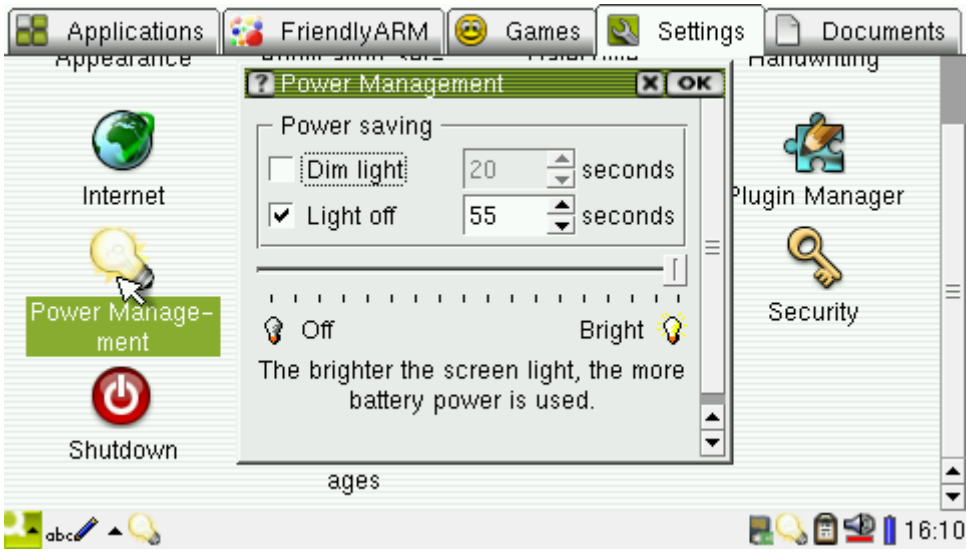
Auto-loop loops automatically. Executing it presents “red”, “yellow”, “white”, “sky blue”, “dark blue”, “green”, “pink” and “black”. During the loop clicking on any place on the screen will return

1.26 Backlight Control

Note: this feature requires an LCD driven by the 1-wire precise touch driver.

If you already played our Mini6410 system you may notice that after power on the board will turn “dark” without being touched for a period. This is a default system action controlled by the backlight management. In the “Settings” tab clicking on “Power Management” will start this utility





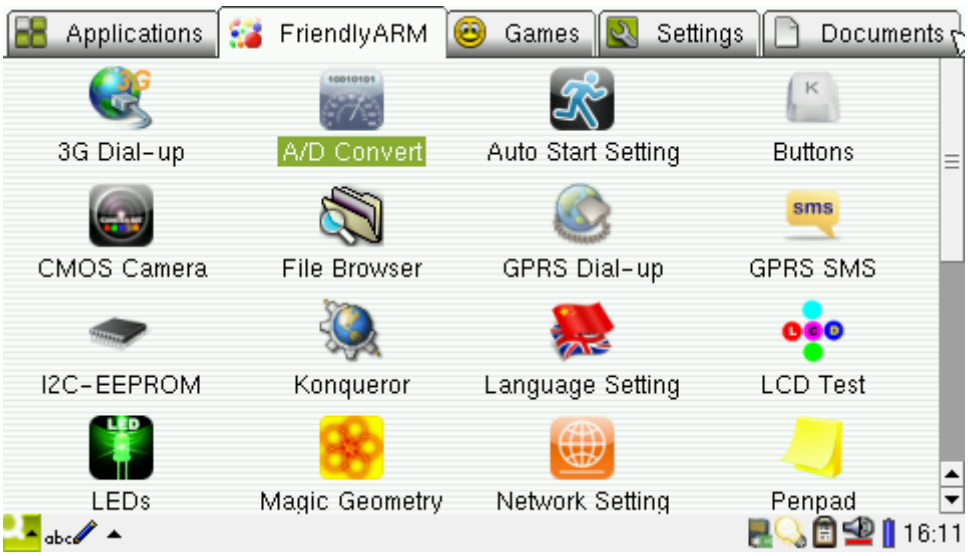
Here the default setting is 25 seconds you can click on the “Up” or “down” arrow to adjust it. If you uncheck “Light off”, the backlight will be on as long as the system is powered on. An LCD driven by the 1-wire precise touch driver integrates the function of adjusting the backlight therefore you can slide the slider to get your desired backlight. When you check “Dim light” you will observe that the light is off gradually. Actually adjusting the backlight in our software is pretty straightforward. You can refer to 2.10 for more details on how to

adjust it via the command line utility.

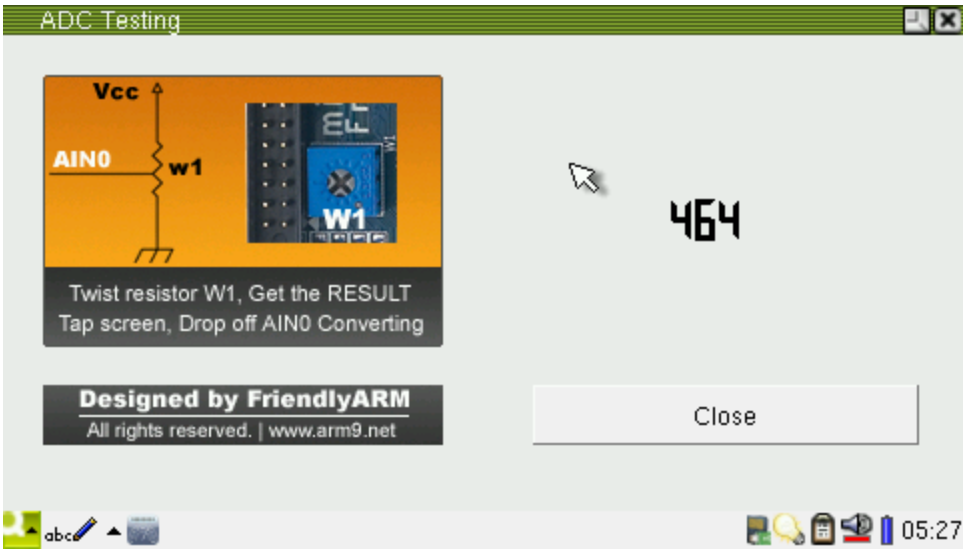
1.27 A/D Conversion

The Samsung 6410 chip has 8 A/D conversion channels but only one converter. In general, AIN4, AIN5, AIN6 and AIN7 are used as YM, YP, XM and XP channels via a four wire resistor. We extended AINs 1-3 which reside on CON6. 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:

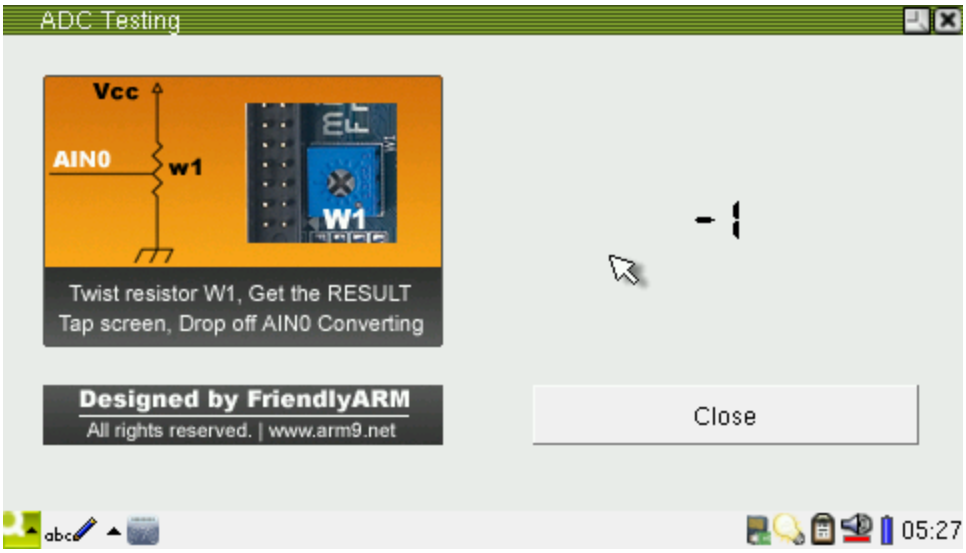
Click on the “ADC Testing” icon in the “FriendlyARM” tab:



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.

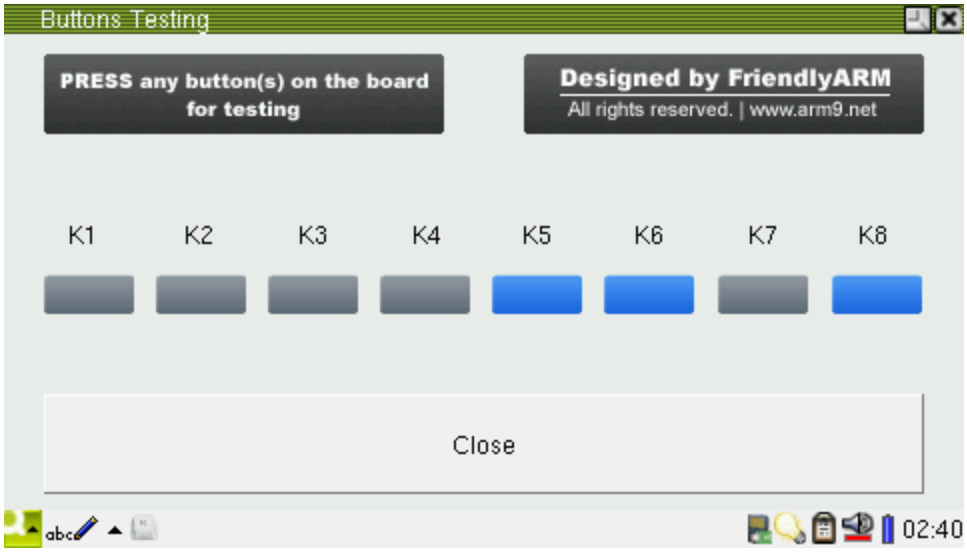


When you click on the touch screen, the A/D converter will take the touch screen as the channel, you will see the result "-1"; when you move your touch pen away from the screen, the A/D converter will take AIN0 as the channel again.



1.28 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.

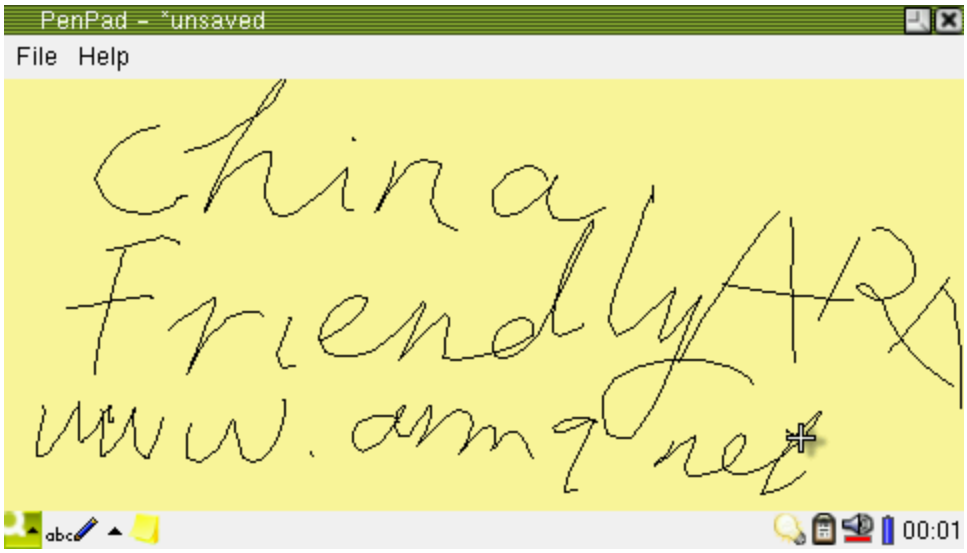


1.29 Touch Pen Test

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, 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.

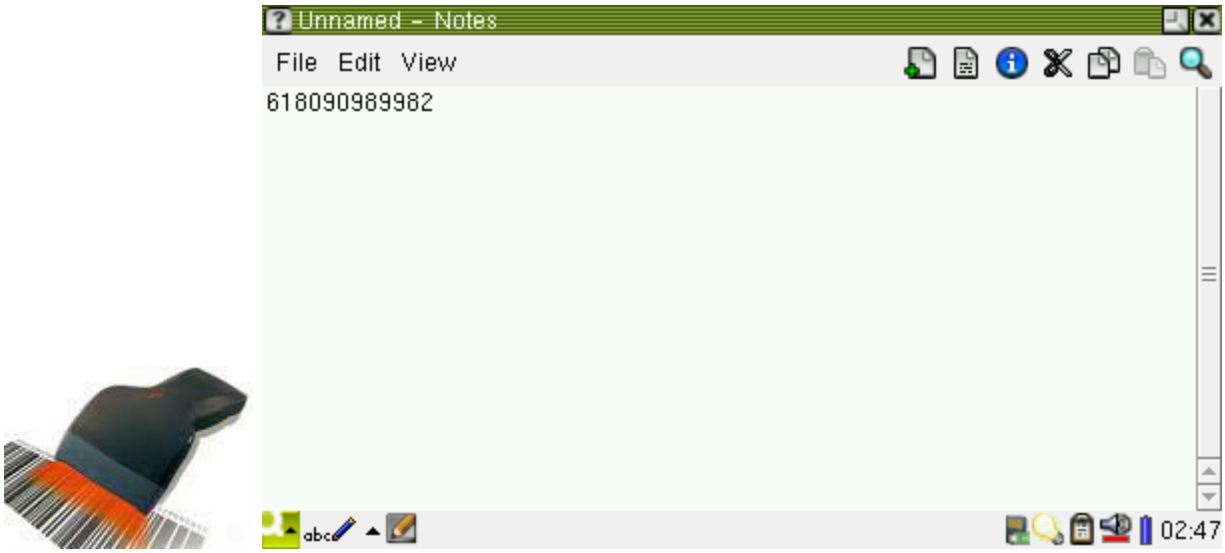


1.30 Barcode Scanning

Our system supports USB barcode scanners which are actually a HID device very similar to a USB keyboard. Therefore a barcode scanner can work any where a USB keyboard works.

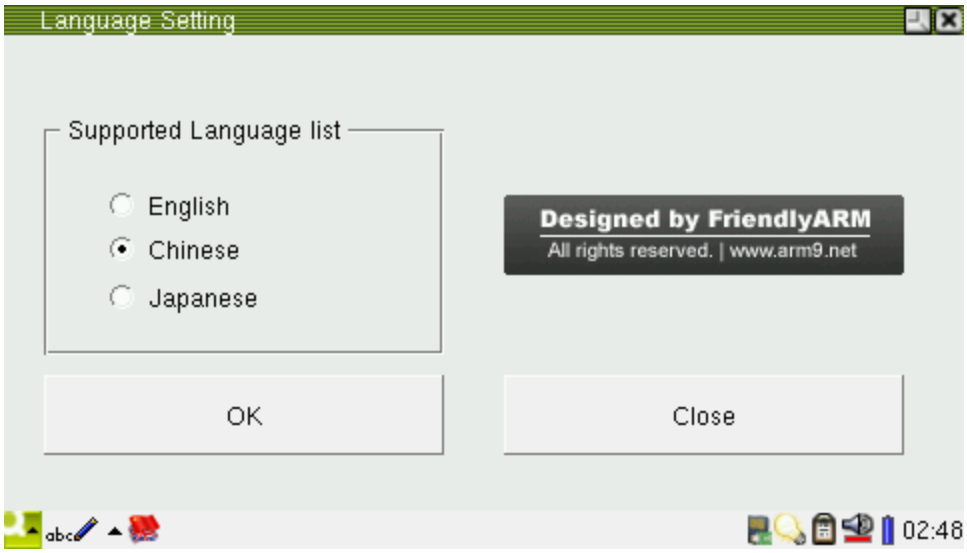
Note: before start this utility, please make sure to plug in your scanner.

Click on the “text editor” icon in the “Application Programs” subgroup, scan a code with your scanner, then you will see the code number displayed in the editor.

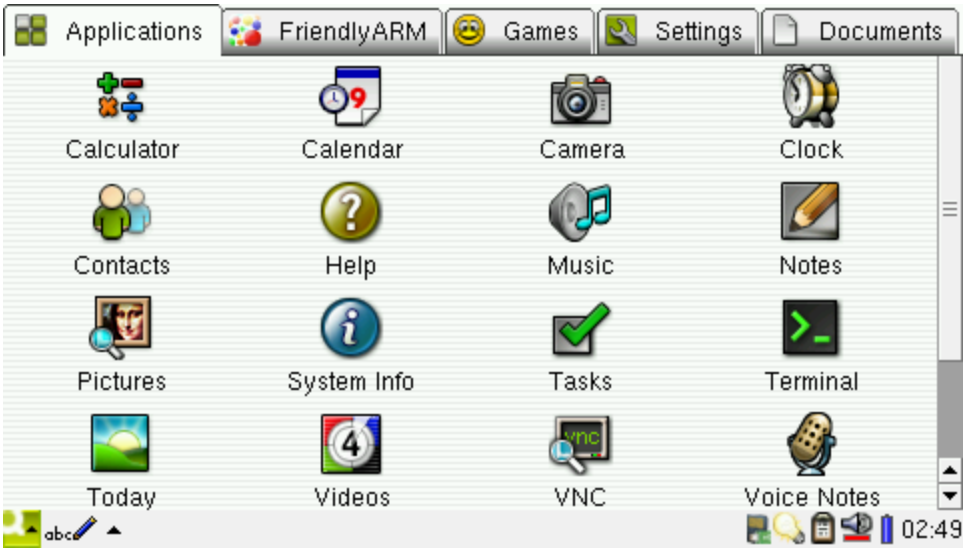


1.31 Language Setting

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).

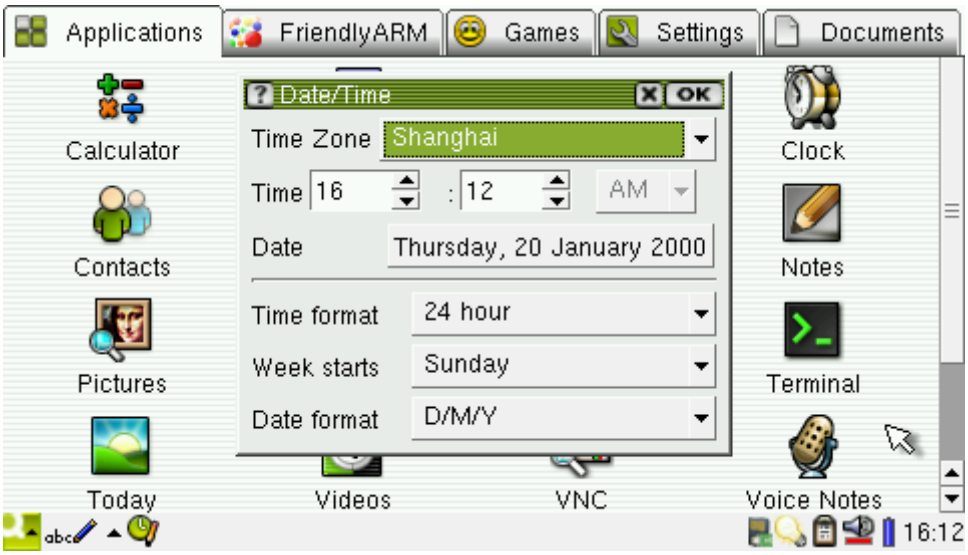


It now supports three languages: English, Chinese and Japanese. When you select “English”, then click on “OK”, a message will popup asking you if you want to change your language setting. Clicking on “Yes” Qtopia will reboot; clicking on “No” it will return. (Note: the Chinese and Japanese versions only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

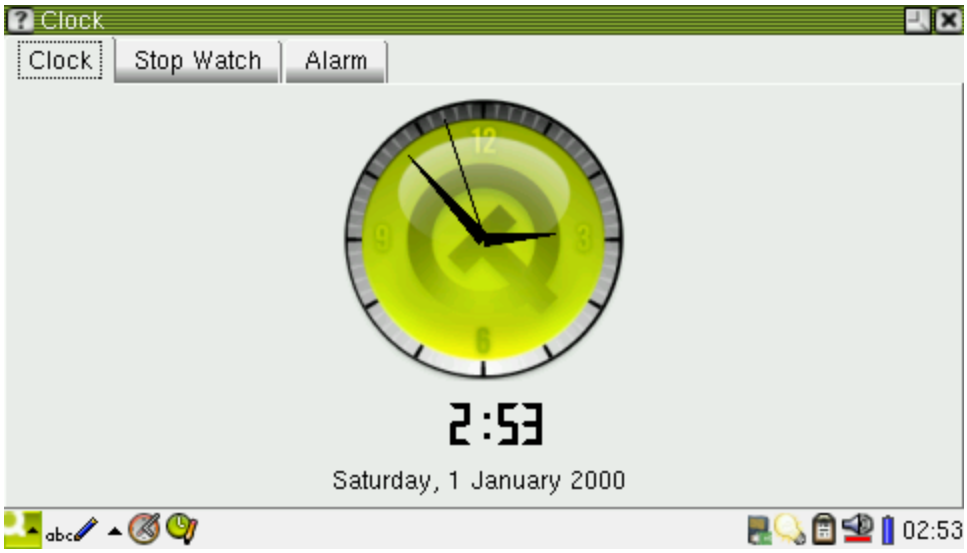


1.32 Set up Time Zone, Date, Time and Alarm Clock

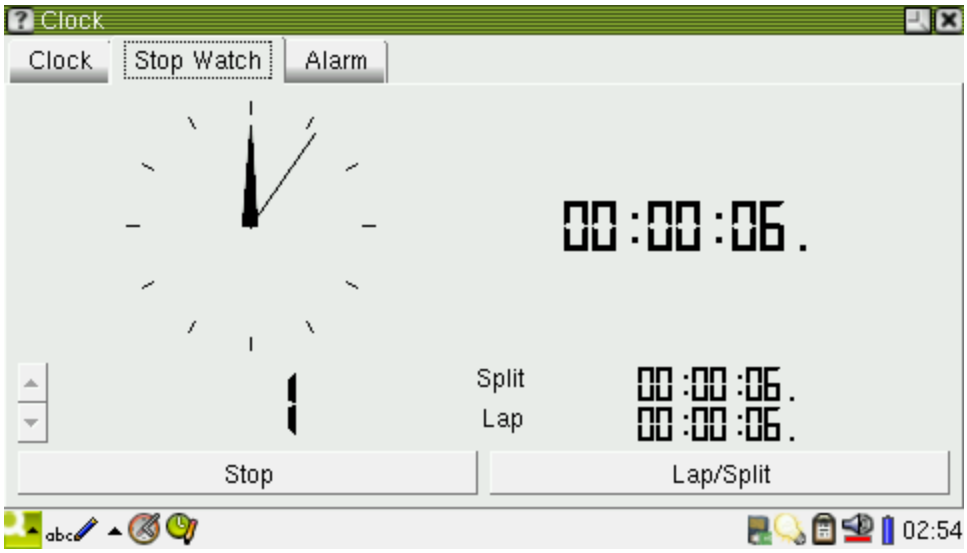
When you get our system, the date and time usually might not be accurate. You can adjust them by yourself. Because the CPU has its own RTC and the board has a backup battery, after you adjust the date and time, they will be saved. To adjust them, click on the time zone area at the right bottom of the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



Select “Clock” from the menu.



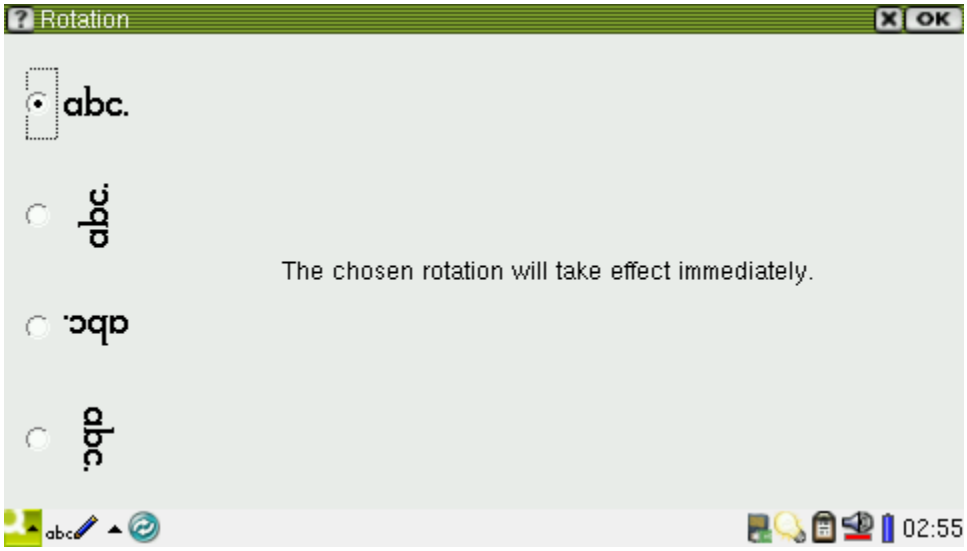
click on “Stop Watch” to open a stopwatch utility



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.

1.33 Rotate Screen

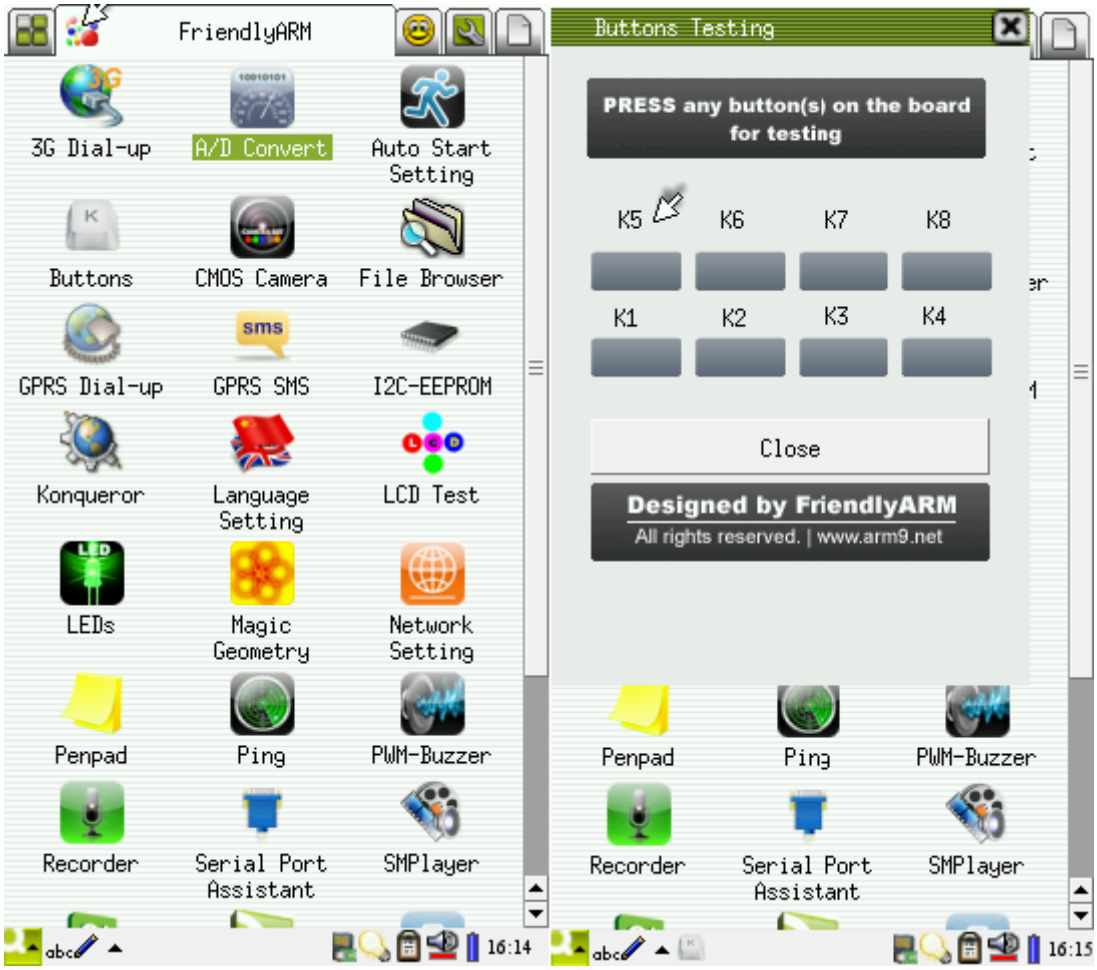
Click on the “rotation” icon in the “settings” tab to enter its interface. You can rotate the screen in four directions.



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 hasn’t made any change to it. In addition the rotation effect is implemented via Qtopia software and has nothing to do with LCD drivers.

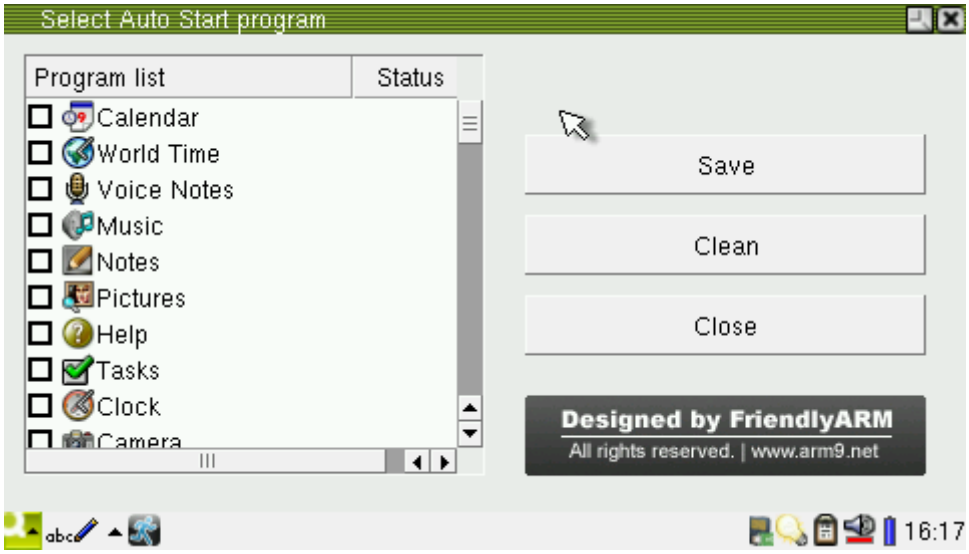
After rotation you will notice that all “FriendlyARM” utilities get rotated too. We implemented this feature to make our utilities displayed properly with different LCDs



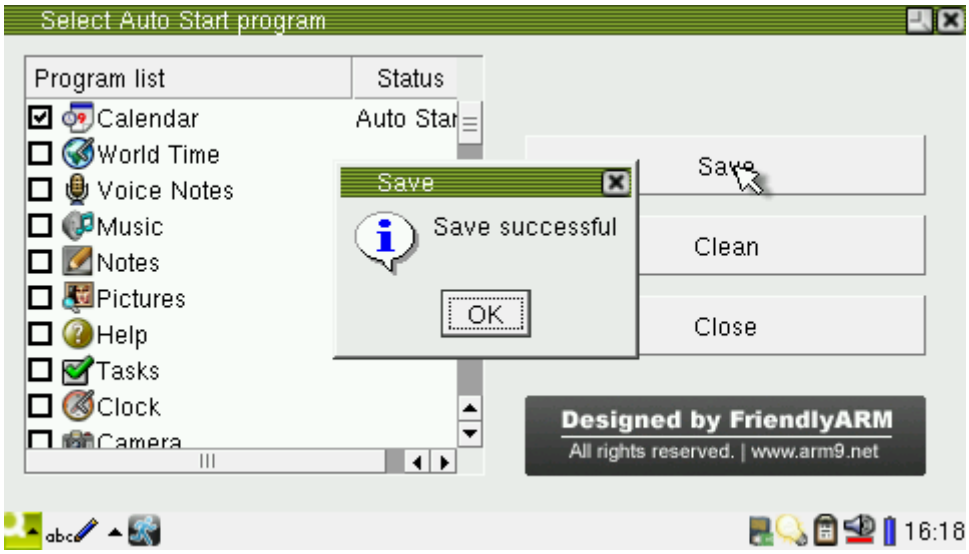
1.34 Set up Auto Run Programs

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”.

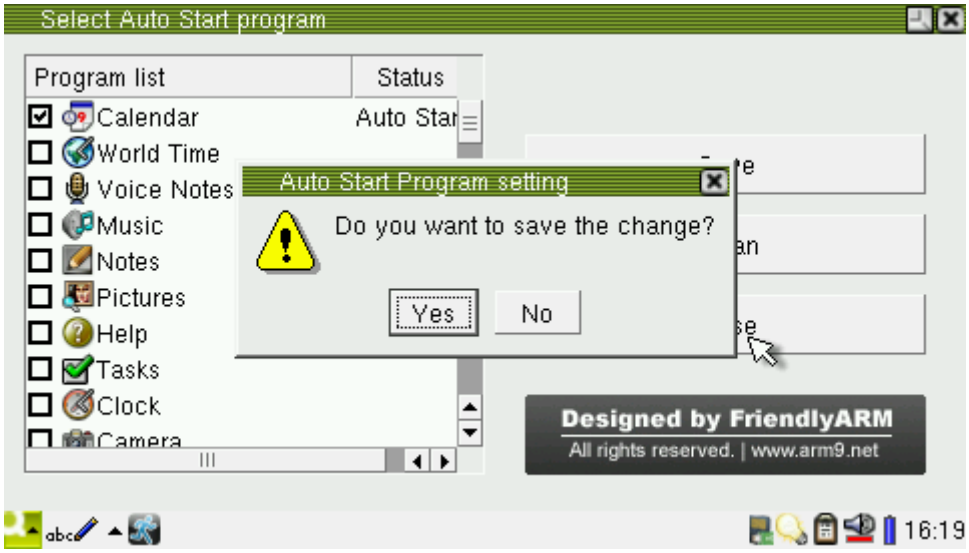
Click on the “Auto Start Setting” icon in the “FriendlyARM” tab.



Those program listed are available programs which include 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.



1.35 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.

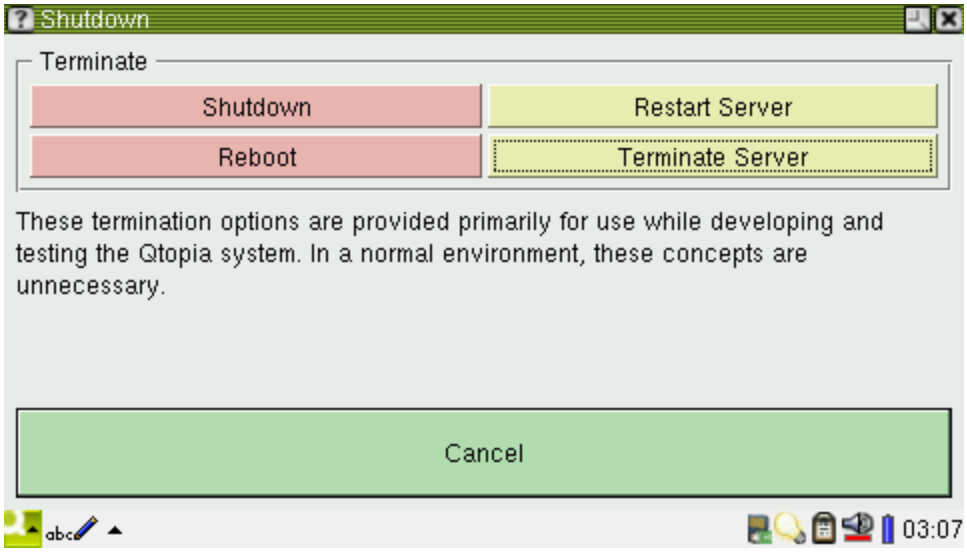
Reboot: This is a “hot” reboot button. If your system boots from the Nor Flash, after you press this button, the system will shutdown, reboot and enter the supervivi main menu. If

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.

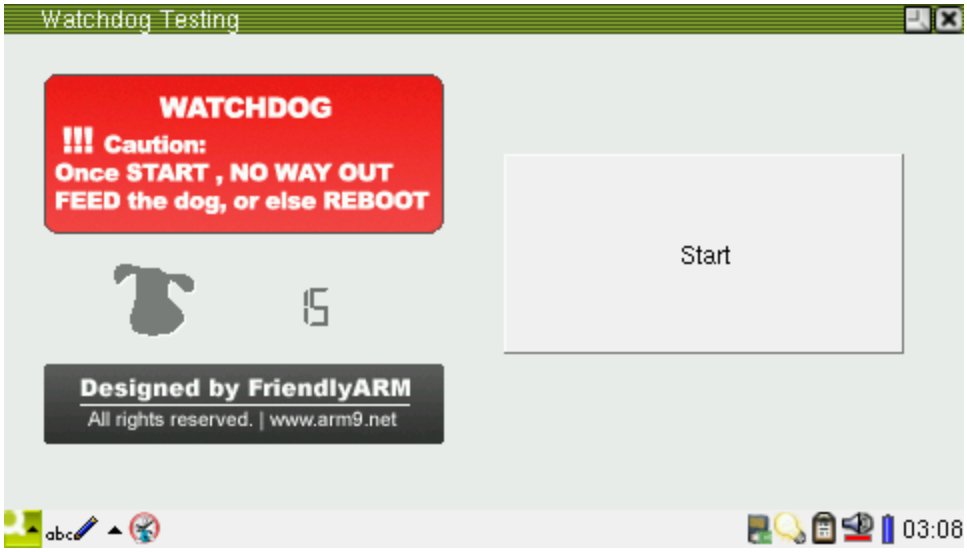


Note: the original Qtopia 2.2.0 system doesn't "shutdown" or "reboot" effectively, we changed its code to make it work.

1.36 Watchdog

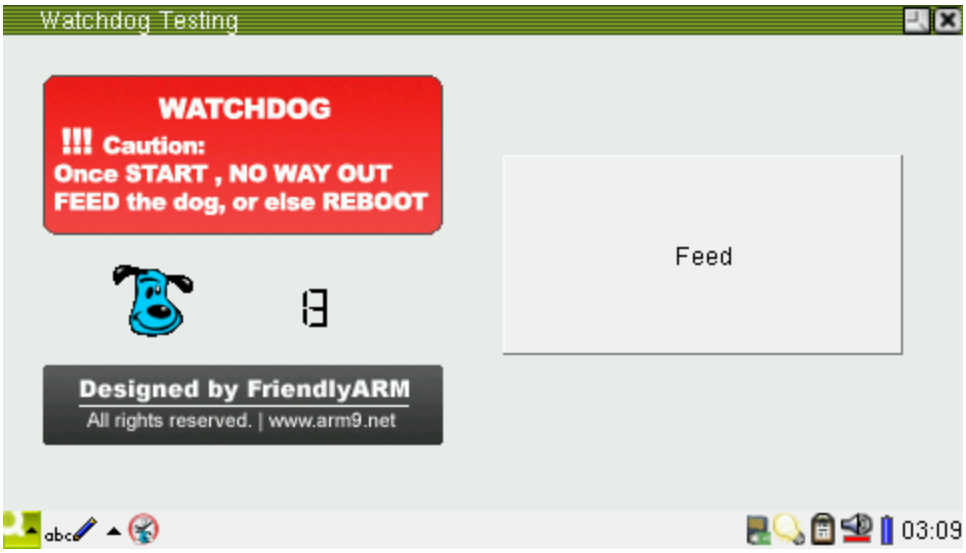
The “Watchdog” is a very basic utility in embedded systems. The S3C6410 chip already has a watchdog. The latest Linux kernel has drivers for it.

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



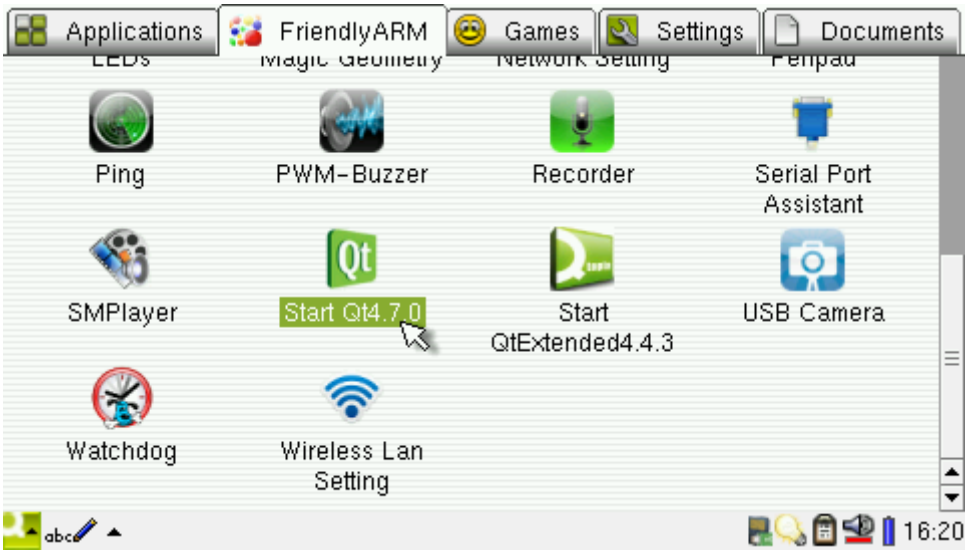
Note: before take any action, please read the notes in the red area: once start, no way out, feed the dog, or else reboot!

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.

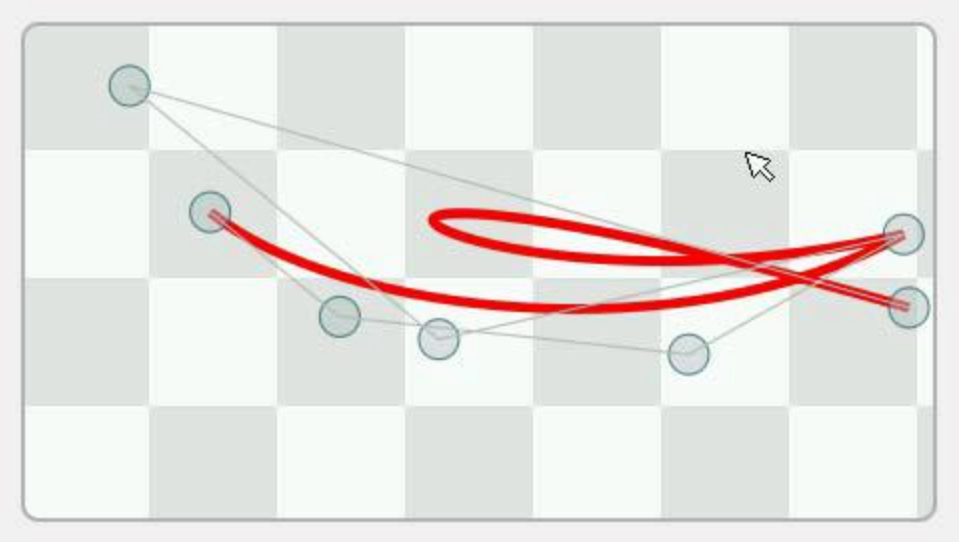


1.37 Start QtE-4.7.0

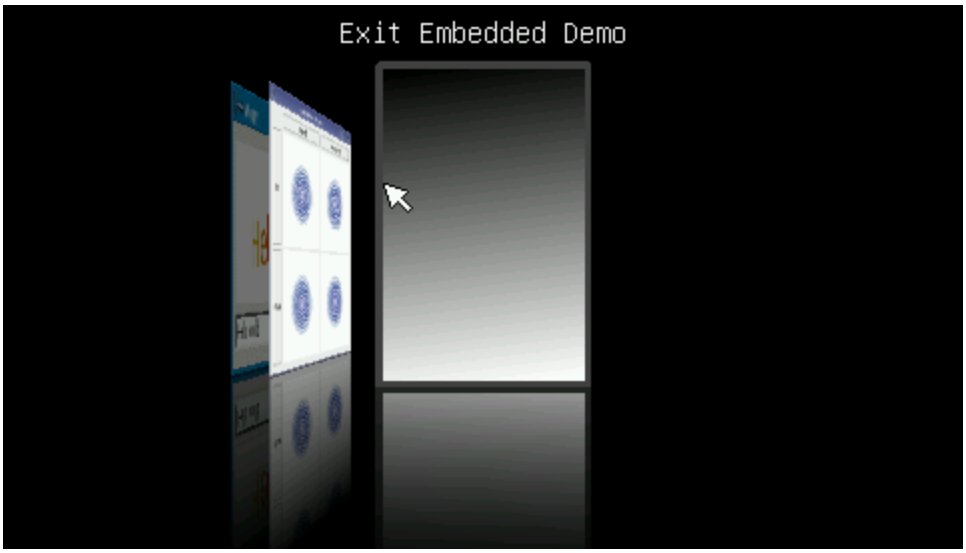
In order for users to switch freely and smoothly between different systems we implemented a feature that allows Qtopia-2.2.0 and QtE-4.7.0 to co-exist in the same file system. In Qtopia-2.2.0, by clicking on a common application icon users will be able to start QtE-4.7.0. After close the QtE-4.7.0 utility, users will be able to return to Qtopia-2.2.0.



QtE-4.7.0 runs as follows. It is a program manager that display a CoverFlow effect. You can drag it left and right and run it by clicking on one of the Covers.



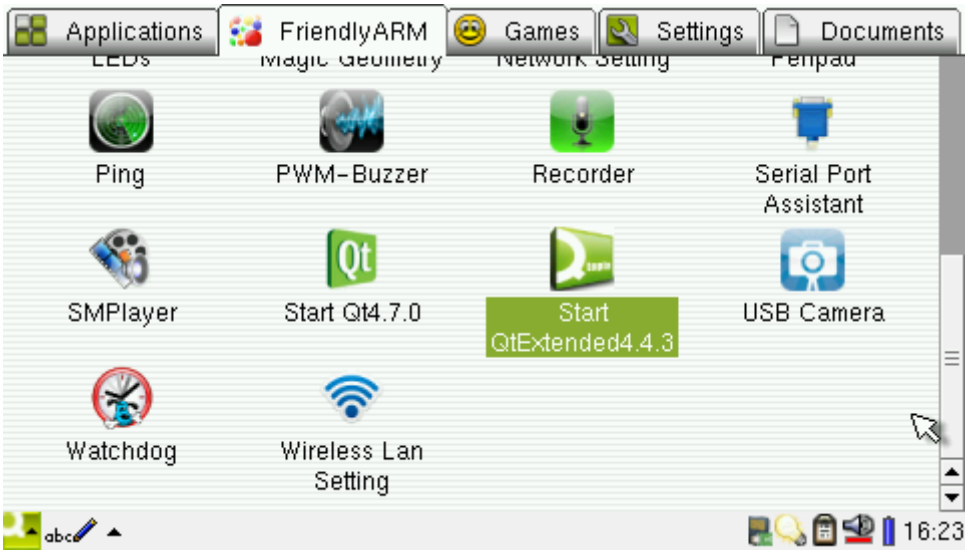
You can exit QtE-4.7.0 by clicking on “Exit Embedded Demo” and return to Qtopia-2.2.0



1.38 Start Qtopia4

In order for users to switch freely and smoothly between different systems we implemented a feature that allows Qtopia-2.2.0 and Qtopia4(Qt Extended 4.4.3 Phone) 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 Qtopia4. After close the Qtopia4 utility, users will be able to return to Qtopia-2.2.0



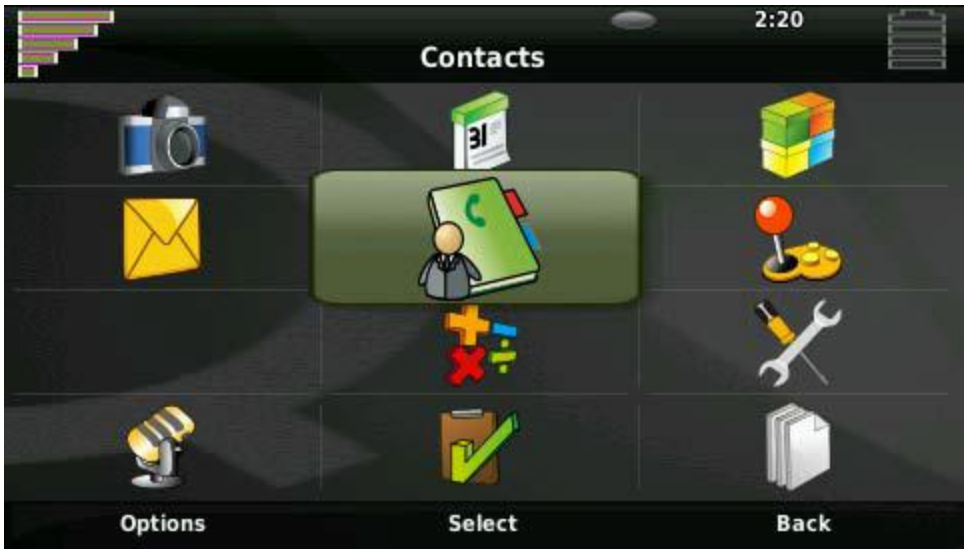
If you have never run Qtopia4 on the system you will see the following GUI after start it

Click on the screen you will be directed to a configuration window where you can set up

your date and time. You can ignore it here and click on “Finish” to continue.

After a while you will enter Qtopia4 and the title is “Qt Extended”

There are three buttons “Options”, “Menu” and “Quit”. “Quit” is implemented by FriendlyARM to allow users easily returning to Qtopia2. You can add your own features too in the source code. Here please click on “Menu” to enter the main function menu.



Qtopia4's applications are very limited which we will not talk too much about here. Click on "Back" to return to the main menu and click on "Quit" to return to Qtopia2.

Note: when quitting users will see a flash which doesn't exist in Qtopia and is implemented by FriendlyARM. It is open source and users can check it.

So far, we have introduced most of the GUI utilities that will be used to manipulate hardware. There are other utilities you can try by yourself.

1.39 Which Qt to Choose

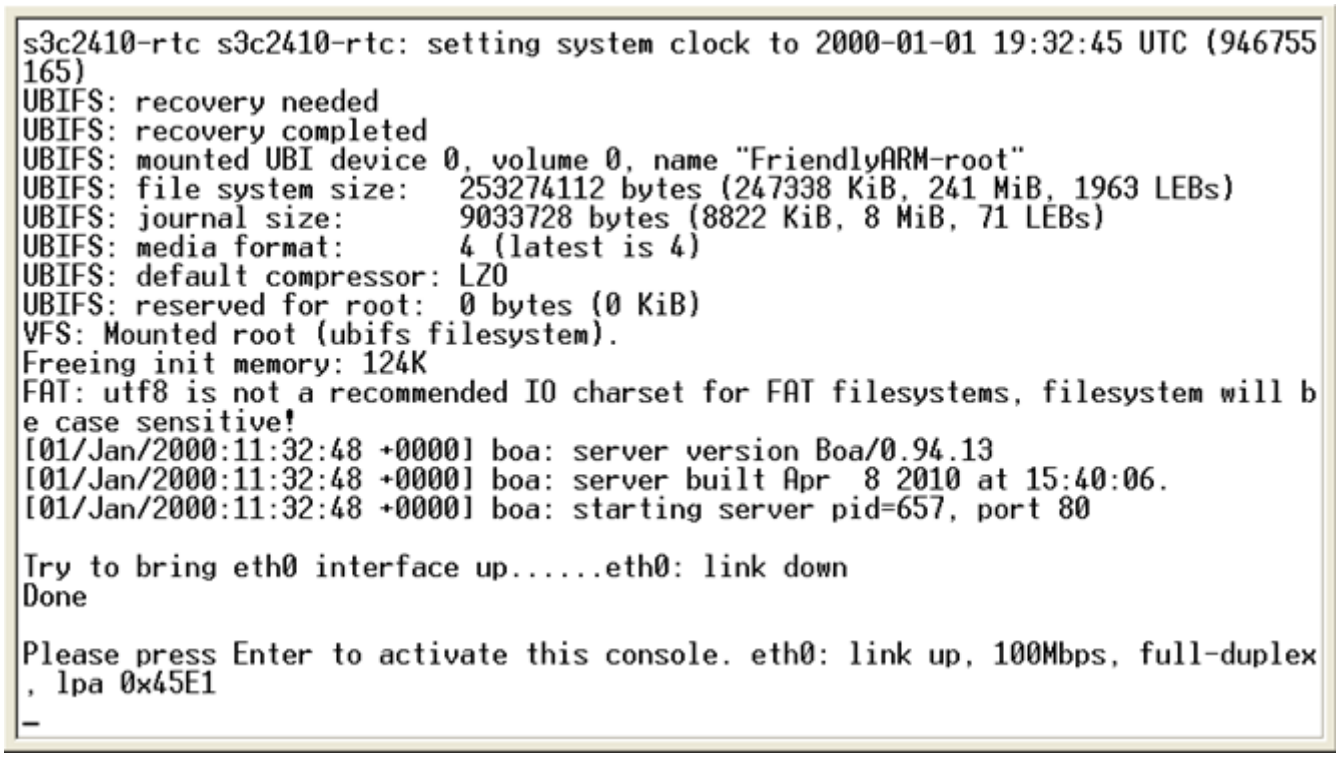
With so many Qt options users may be confused about which one to use. Actually it all depends on what you need. For development boards it would be better to have a complete desktop version (Qtopia is one for mobile devices) for various LCDs. Per this requirement we took Qtopia-2.2.0 and made it possible for Qtopia4 and QtE-4.7.0 to co-exist and allow users to smoothly switch between them. It is not a fancy technology and we just made it based on basic C/C++ functions which are enough for us to achieve what we need.

If you don’t need the whole system and just some of the applications you recommend you to choose QtE-4.7 or high versions since they can work in more platforms and are easier for beginners to learn and migrate. In addition a QtE-4.7 application doesn’t take too much memory.

2 Play Mini6410 via Command Line

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.



2.1 Play MP3

This section will give a brief introduction on how to run Linux commands and various

application programs in Linux via a super terminal. Before move forward, please connect your board with a PC and start a super terminal. The following screenshot is what you might see after you set up your super terminal and connection with your board

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.

```
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will b
e case sensitive!
[01/Jan/2000:11:32:48 +0000] boa: server version Boa/0.94.13
[01/Jan/2000:11:32:48 +0000] boa: server built Apr  8 2010 at 15:40:06.
[01/Jan/2000:11:32:48 +0000] boa: starting server pid=657, port 80

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

Please press Enter to activate this console. eth0: link up, 100Mbps, full-duplex
, lpa 0x45E1

[root@FriendlyARM /]#
[root@FriendlyARM /]# madplay /root/Documents/viva-la-vida.mp3
MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-2004 Robert Leslie et al.
          Title: Viva La Vida
         Artist: Coldplay
          Album: Viva La Vida Or Death And All His Friends
          Track: 7
           Year: 2008
          Genre: Rock
        Encoder: iTunes v7.6
        Comment: Rip & Rls bY Team COC
```

In the latest Linux kernel, we integrated a driver for ALSA audio interface. The madplay utility plays audio files via this interface too and related ALSA libraries are also integrated into the system.

2.2 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.

2.3 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.

2.4 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.

```
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 /]# ls sdcard/
??                                  logo_linux_clut224.png
linux-2.6.29.fa-src-2009-03-24.tar.gz  zImage_29.bin
[root@FriendlyARM /]# 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/sdcard on /sdcard type vfat (rw,sync,nosuid,nodev,noatime,nodiratime,fmas
0022,dmask=0022,codepage=cp437,iocharset=iso8859-1)
[root@FriendlyARM /]# _
```

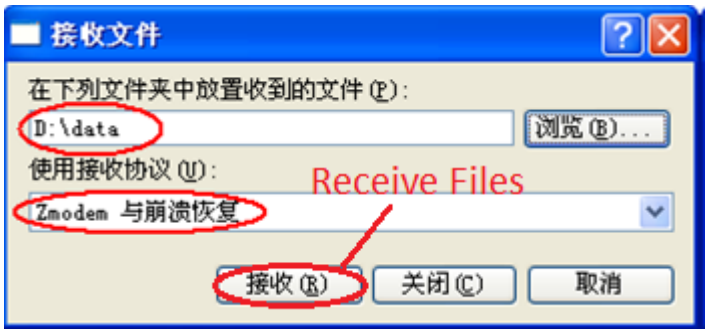
2.5 File Transfers to and from PC via Serial Port

Note: some users may not get file transfers done successfully via a USB to Serial connector it could result from the cable's bad quality

After login into the Mini6410 system via a serial port, you can transfer files to and from a host PC by using command "**rz**"or "**sz**" as follows:

(1) Transfering files by using "sz"

Open a super terminal, click on the mouse's right button, then click on "Receive files" to set up the destination directory and the protocol this transfer will use, see the screenshot below:



Type "**sz /root/Documents/viva-la-vida.mp3**" in the shell to transfer the "viva-la-vida.mp3" file under the "/root/Documents" directory to the host PC. It took a while to transfer this big file. After it is done, the system will save it in the directory you set in the previous step. Please see the screen-shot below:

(2) Transfering files by using "rz"

In your Mini6410 system, type "rz" to receive files from a host PC.

Open a super terminal, click on the mouse's right button, select "Send file", set up the file being sent and the protocol the transfer will use. Then send the file:

Click on "Send" and the board will begin to receive your file

After the transfer is done, the current directory will get this file. You can verify it by using "md5sum" to check whether this file is the same as the original one.

2.6 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 a 0.2 second in between.

#echo 1 0.2 >/tmp/led-control

After this command is executed 4 LEDs will be running the accumulator one by one with a 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

2.7 User Button Test

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

2.8 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:

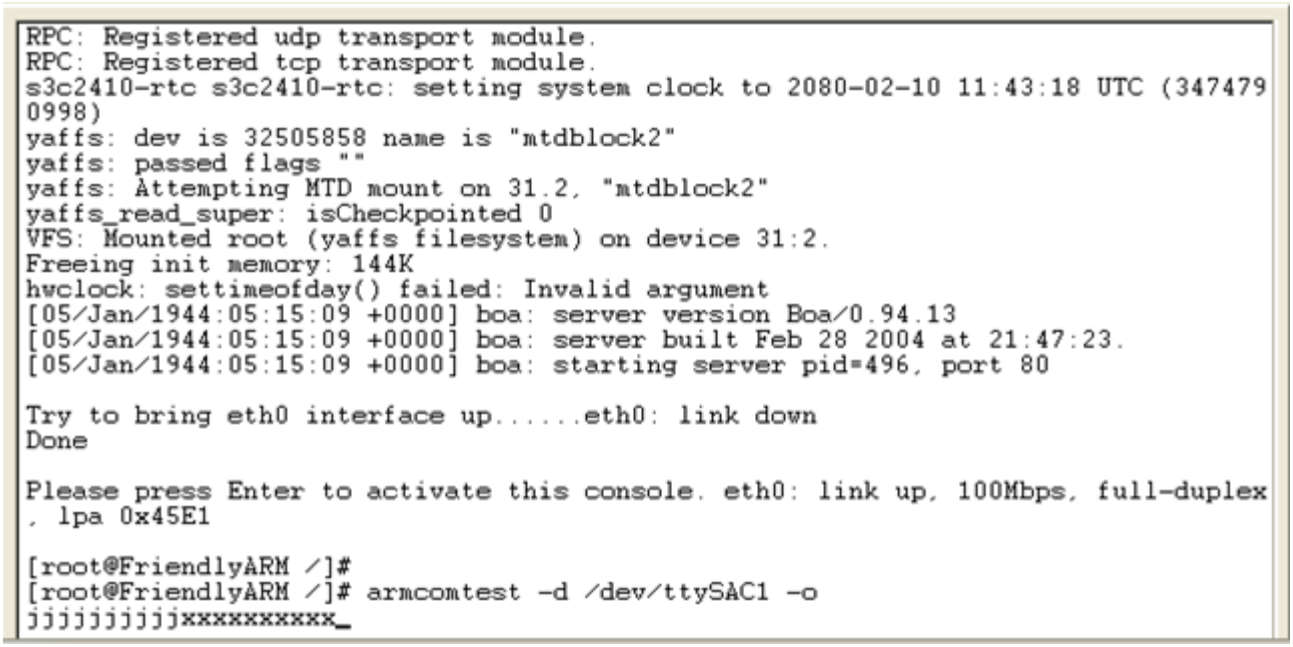
#armcomtest –d /dev/ttySAC1 -o

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:

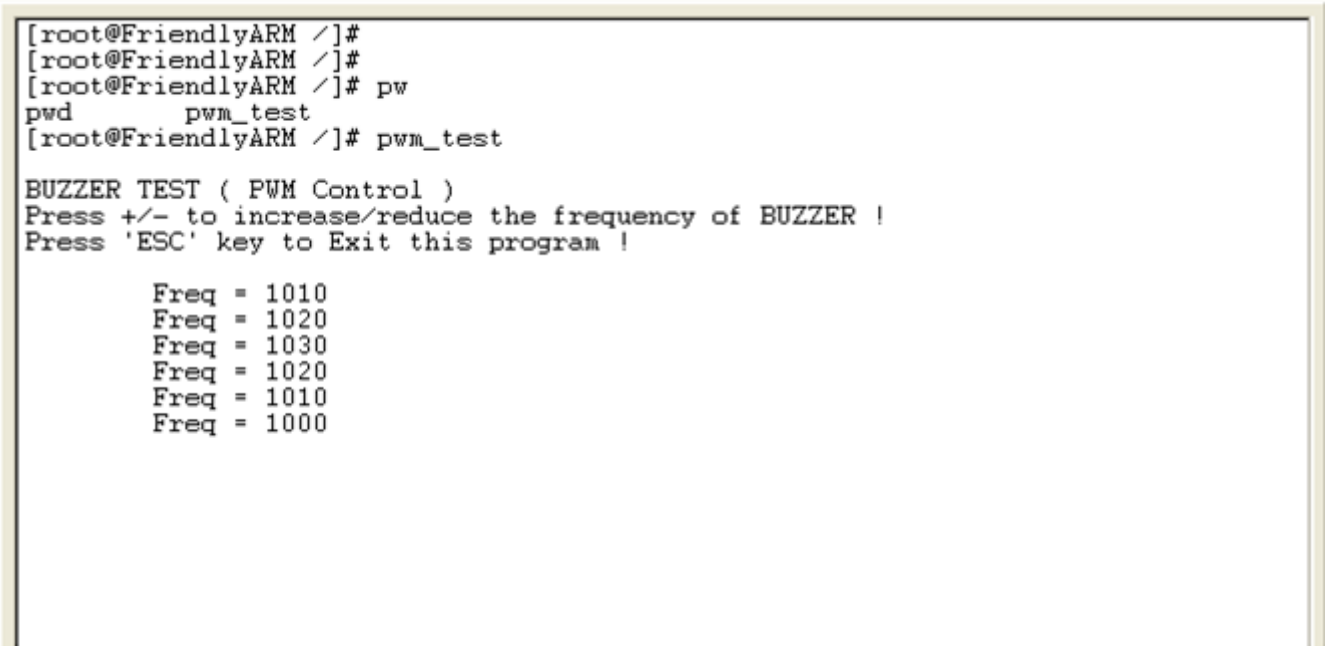
#armcomtest –d /dev/ttySAC2 -o

Here is a screenshot



2.9 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.



2.10 Backlight Control

Note: the device file for the LCD backlight is /dev/backlight-1wire.

We also implement the feature of adjusting backlight for 1-wire precise touch LCDs. It supports up to 127 levels of adjustment. To turn off the backlight you can feed “0” to the device file as follows:

Type the command: echo 0 > /dev/backlight

When feed 1-127 to the device file you will observe different levels of light. 127 is the highest.In general 15 will begin to show some light. 1-15 makes the screen completely dark. Values higher than 127 will be treated as 127.

Try: echo 15 > /dev/backlight you will be able to see some light.

2.11 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 /]#
```

2.12 AD Conversion

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

2.13 TV-OUT Test

Enter "**/usr/bin**" (attention: NOT "/user/bin"), run "**tv-test**" to begin TV-OUT testing, it will play "/usr/bin/TestVectors/wanted.264" and output it to your connected TV:

#cd /usr/bin

#tv-test

2.14 Multi-Media Test

Please refer to 1.3

2.15 Test USB WiFi or SD WiFi

For users to utilize USB WiFi or SD WiFi cards we originally developed a command set:USB WiFi kits for the Mini2440 system. This command set supports up to one thousand various USB WiFi cards (most of which utilize similar chips). We also integrated this command set into our Mini6410 system and also integrated the SD WiFi driver.

This command set includes a WiFi driver and three commands:

- scan-wifi – scans nearby wireless networks
- start-wifi – starts connecting to a wireless network
- stop-wifi – closes a wireless connection

All of them are under the "/usr/sbin" directory

1. Scanning Nearby Wireless Networks

Note: the following examples were tested on TL-WN321G+. The SD-WiFi works very similar to this. We leave it for users to explore.

Connect your USB WiFi card to your board you will see the following screenshot

```
lib/modules/2.6.32.2-FriendlyARM/net/wireless/cfg80211.ko
lib/modules/2.6.32.2-FriendlyARM/net/mac80211/
lib/modules/2.6.32.2-FriendlyARM/net/mac80211/mac80211.ko
lib/firmware/
lib/firmware/rt73.bin
lib/firmware/ar9271.fw
usr/
usr/sbin/
usr/sbin/start-wifi
usr/sbin/wpa_supplicant
usr/sbin/stop-wifi
usr/sbin/scan-wifi
usr/share/
usr/share/udhcpc/
usr/share/udhcpc/default.script
[root@FriendlyARM /]# usb 1-1: new full speed USB device using s3c2410-ohci and
address 2
usb 1-1: New USB device found, idVendor=148f, idProduct=2573
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
usb 1-1: Product: 54M.USB.......
usb 1-1: Manufacturer: Ralink
usb 1-1: configuration #1 chosen from 1 choice

[root@FriendlyARM /]# _
```

Execute the scan command to search for a network: #scan-wifi

```
usr/sbin/scan-wifi
usr/share/
usr/share/udhcpc/
usr/share/udhcpc/default.script
[root@FriendlyARM /]# usb 1-1: new full speed USB device using s3c2410-ohci and
address 2
usb 1-1: New USB device found, idVendor=148f, idProduct=2573
usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
usb 1-1: Product: 54M.USB.......
usb 1-1: Manufacturer: Ralink
usb 1-1: configuration #1 chosen from 1 choice

[root@FriendlyARM /]# scan-wifi
cfg80211: Calling CRDA to update world regulatory domain
Registered led device: rt73usb-phy0::radio
Registered led device: rt73usb-phy0::assoc
Registered led device: rt73usb-phy0::quality
usbcore: registered new interface driver rt73usb
usbcore: registered new interface driver ath9k_hif_usb
        63% FriendlyARM (Security)
        37% TP-LINK_65FC92
        34% NETGEAR
3 Access Point Found
[root@FriendlyARM /]# _
```

"63" indicates the strength of the signal

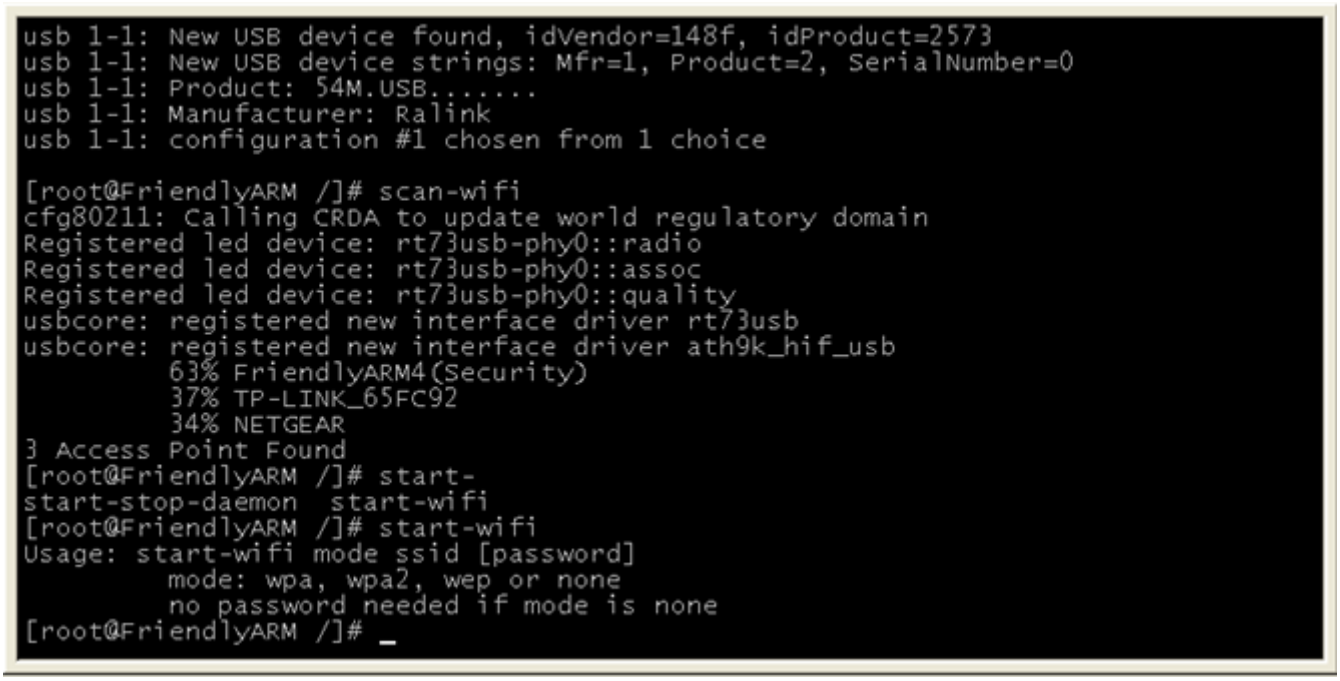
"Security" indicates password is needed

In our example it found three networks and “63%” indicated the strength of the signals.

Those that need passwords will be noted by “Security”.

2. Start Connection

“start-wifi” will connect your board to a specified netowork. After execute “start-wifi” you will see the following information:



mode – security type, such as “wpa”, “wpa2”, “wep” or “none”. “None” means that network doesn’t require a password

ssid – network name such as “FriendlyARM4”, “NETGEAR” shown above.

password – password to log in the network.

We will present two examples: one for a network that doesn’t require a password and the other for a network that does require a password

2.1 Connecting to an Open Network that Doesn’t Require a Password

Step1: Command “scan-wifi” to scan your nearby networks. Here we found “FriendlyARM-Test” which was dedicated for this testing

```
[root@FriendlyARM /]# scan-wifi
cfg80211: Calling CRDA to update world regulatory domain
Registered led device: rt73usb-phy0::radio
Registered led device: rt73usb-phy0::assoc
Registered led device: rt73usb-phy0::quality
usbcore: registered new interface driver rt73usb
usbcore: registered new interface driver ath9k_hif_usb
         63% FriendlyARM4(Security)
         37% TP-LINK_65FC92
         34% NETGEAR
3 Access Point Found
[root@FriendlyARM /]# start-
start-stop-daemon  start-wifi
[root@FriendlyARM /]# start-wifi
Usage: start-wifi mode ssid [password]
         mode: wpa, wpa2, wep or none
         no password needed if mode is none
[root@FriendlyARM /]# scan-wifi
         54% FriendlyARM4(Security)
         34% TP-LINK_65FC92
         37% test engineers(Security)
        100% FriendlyARM-Test
4 Access Point Found
[root@FriendlyARM /]# _
```

Step2: Command “start-wifi none FriendlyARM-Test” to connect to this network

```
[root@FriendlyARM /]# start-
start-stop-daemon  start-wifi
[root@FriendlyARM /]# start-wifi
Usage: start-wifi mode ssid [password]
         mode: wpa, wpa2, wep or none
         no password needed if mode is none
[root@FriendlyARM /]# scan-wifi
         54% FriendlyARM4(Security)
         34% TP-LINK_65FC92
         37% test engineers(Security)
        100% FriendlyARM-Test
4 Access Point Found
[root@FriendlyARM /]# start-wifi none FriendlyARM-Test
udhcpc (v1.13.3) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.3.100...
Lease of 192.168.3.100 obtained, lease time 7200
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.3.1
[root@FriendlyARM /]#
```

Moments later you will notice that your board will be allocated an IP. Here we got "192.168.3.100". "Ping" this network to test your connection.

```
[root@FriendlyARM /]# start-wifi none FriendlyARM-Test
udhcpc (v1.13.3) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.3.100...
Lease of 192.168.3.100 obtained, lease time 7200
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.3.1
[root@FriendlyARM /]# ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=64 time=24.194 ms
64 bytes from 192.168.3.1: seq=1 ttl=64 time=21.053 ms
64 bytes from 192.168.3.1: seq=2 ttl=64 time=20.289 ms
64 bytes from 192.168.3.1: seq=3 ttl=64 time=20.235 ms
64 bytes from 192.168.3.1: seq=4 ttl=64 time=21.180 ms
64 bytes from 192.168.3.1: seq=5 ttl=64 time=20.631 ms
^C
--- 192.168.3.1 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 20.235/21.263/24.194 ms
[root@FriendlyARM /]#
```

Or you can try this IP in your PC's web browser.

2.2 Connect to a Network That Requires a Password

This procedure is very similar to the previous one but you need to know the network's security type and its password:

Step1: Select your network's security type. (The router we used in this example was

TL-WR740N.) Open its web page.

You can choose from the following three:

- WPA-PSK/WPA2-PSK
- WPA/WPA2
- WEP

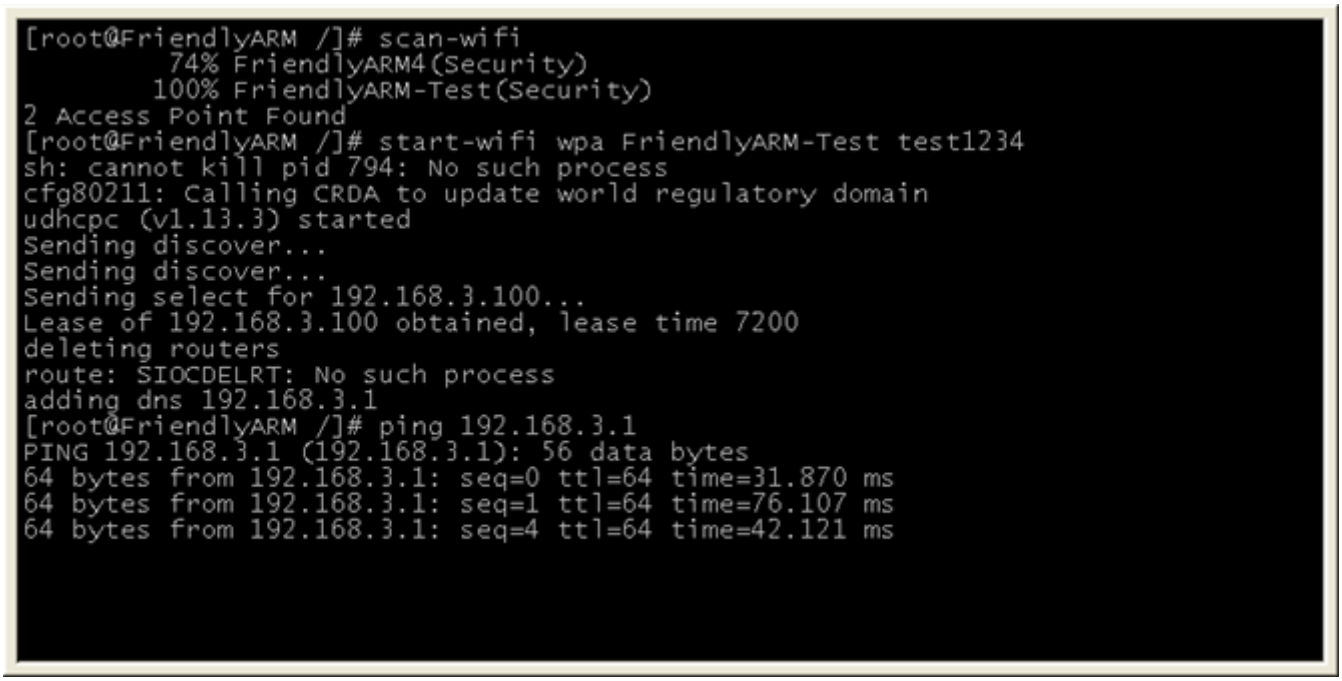
In our example we selected “WPA” and the password was “test1234”. After that click on

"Save" and reboot your router. Note: on how to configure your router you need to refer to your router's manual.

Step2: Command "scan-wifi" to scan your nearby networks. Here we found "FriendlyARM-Test" which was dedicated for this testing

Step3: Command "start-wifi wpa FriendlyARM-Test test1234" to connect to this network.

Moments later you will notice that your board will be allocated an IP. Here we got "192.168.3.100". "Ping" this network to test your connection



Or you can try this IP in your PC's web browser

3. Close USB WiFi Connection

Command "stop-wifi" to close your USB WiFi connection

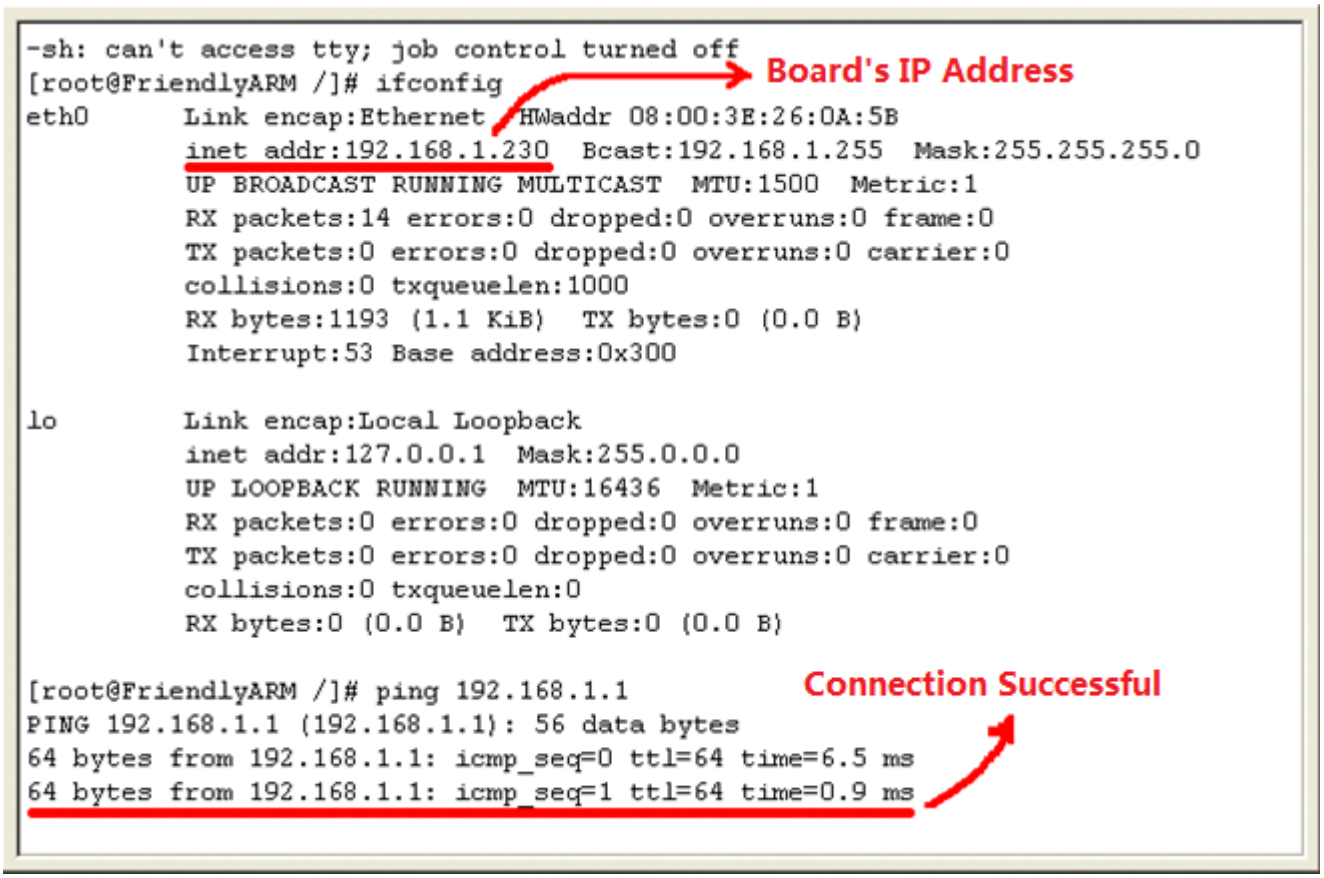
2.16 Preview with CMOS Camera

Connect your CAM130 camera to your board's camera interface, power on, command "camtest" you will be able to preview

2.17 Telnet

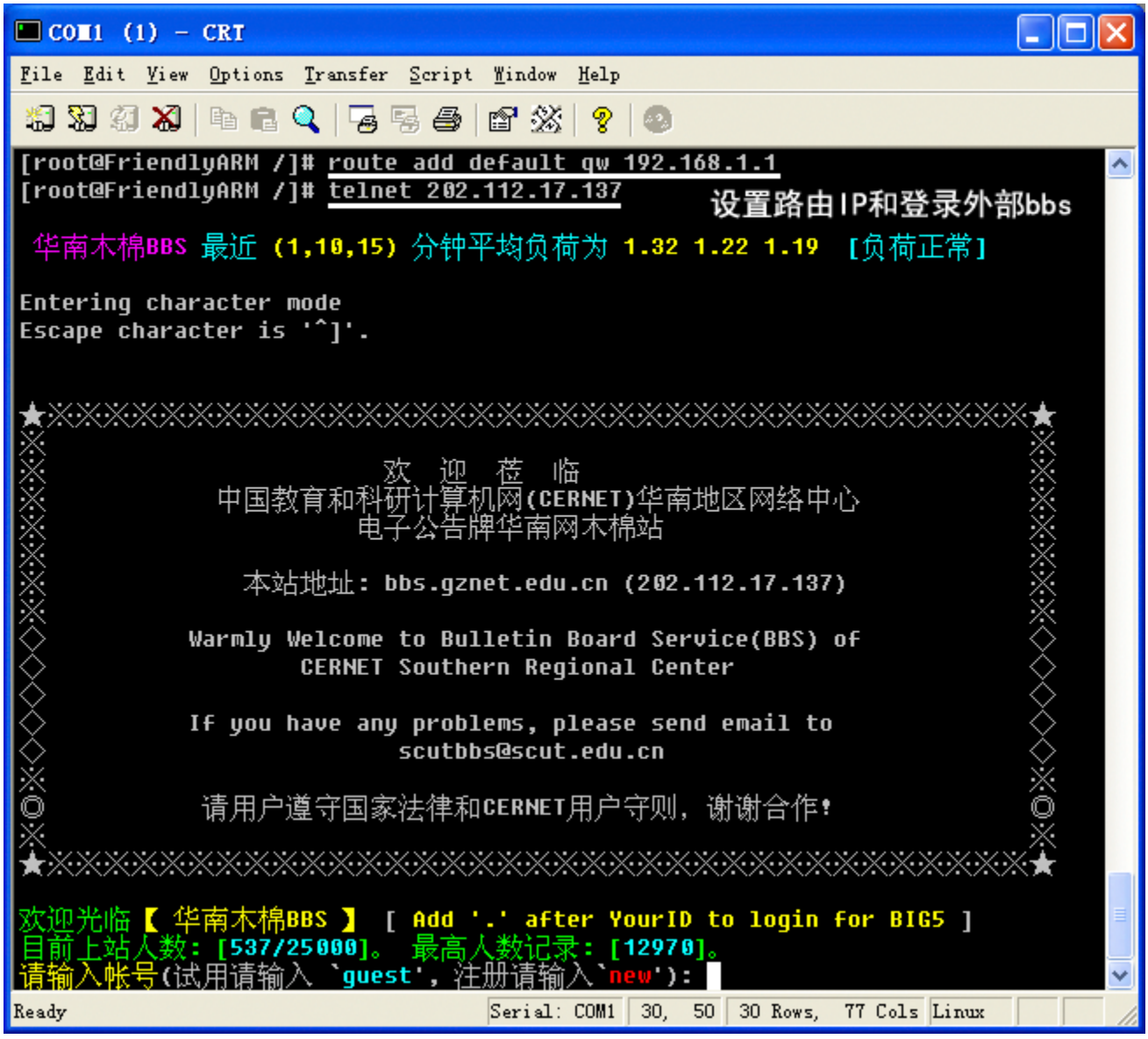
"Telnet" is a popular utility. If your board is connected to the internet you can telnet a bbs.

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".



2.18 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:

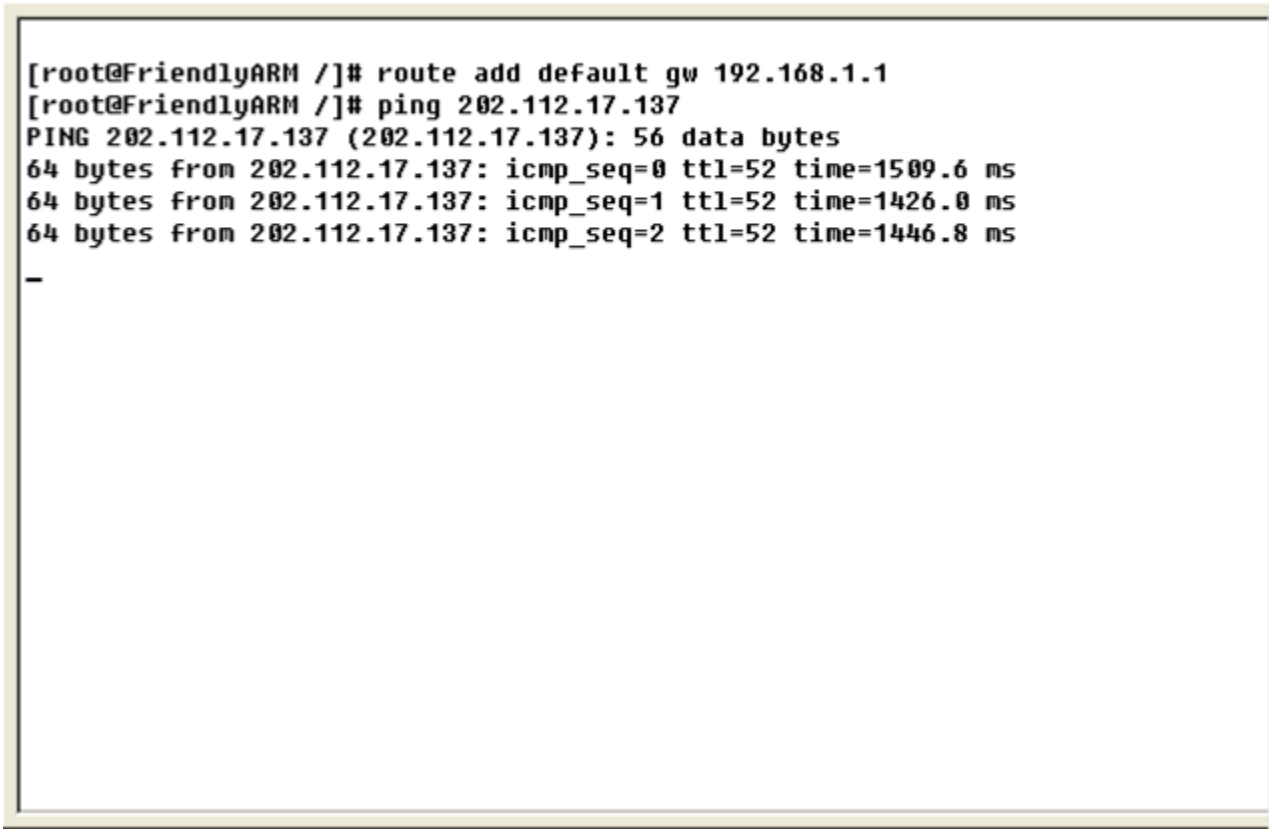
route add default gw 192.168.1.1

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

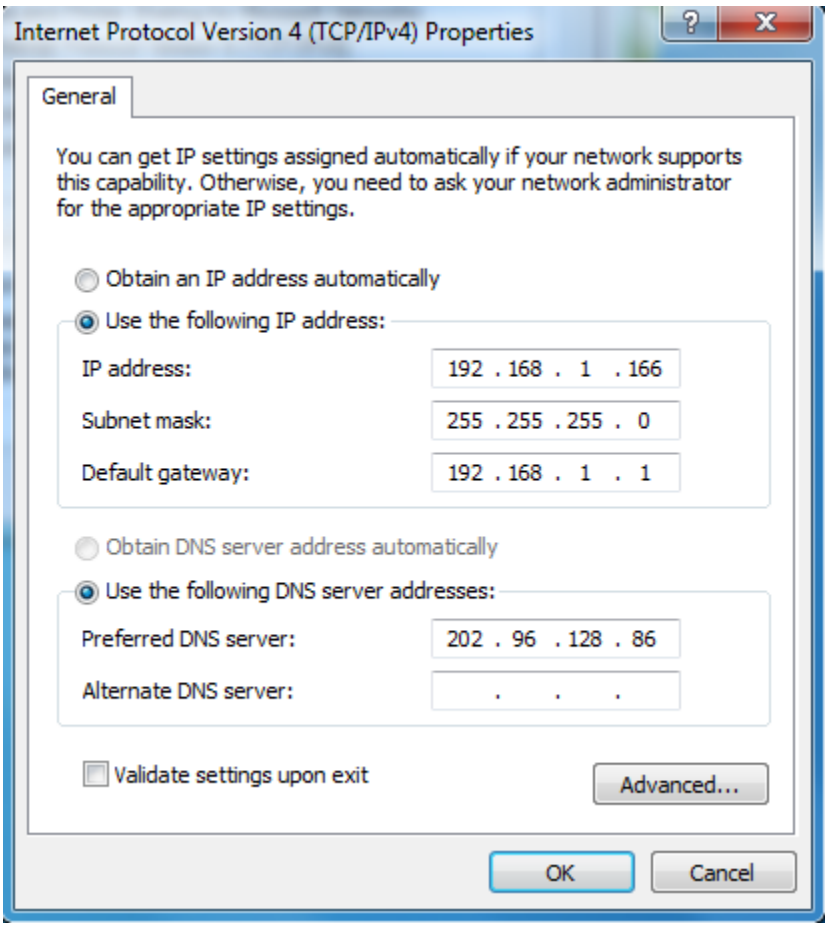
202.112.17.137):

#**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

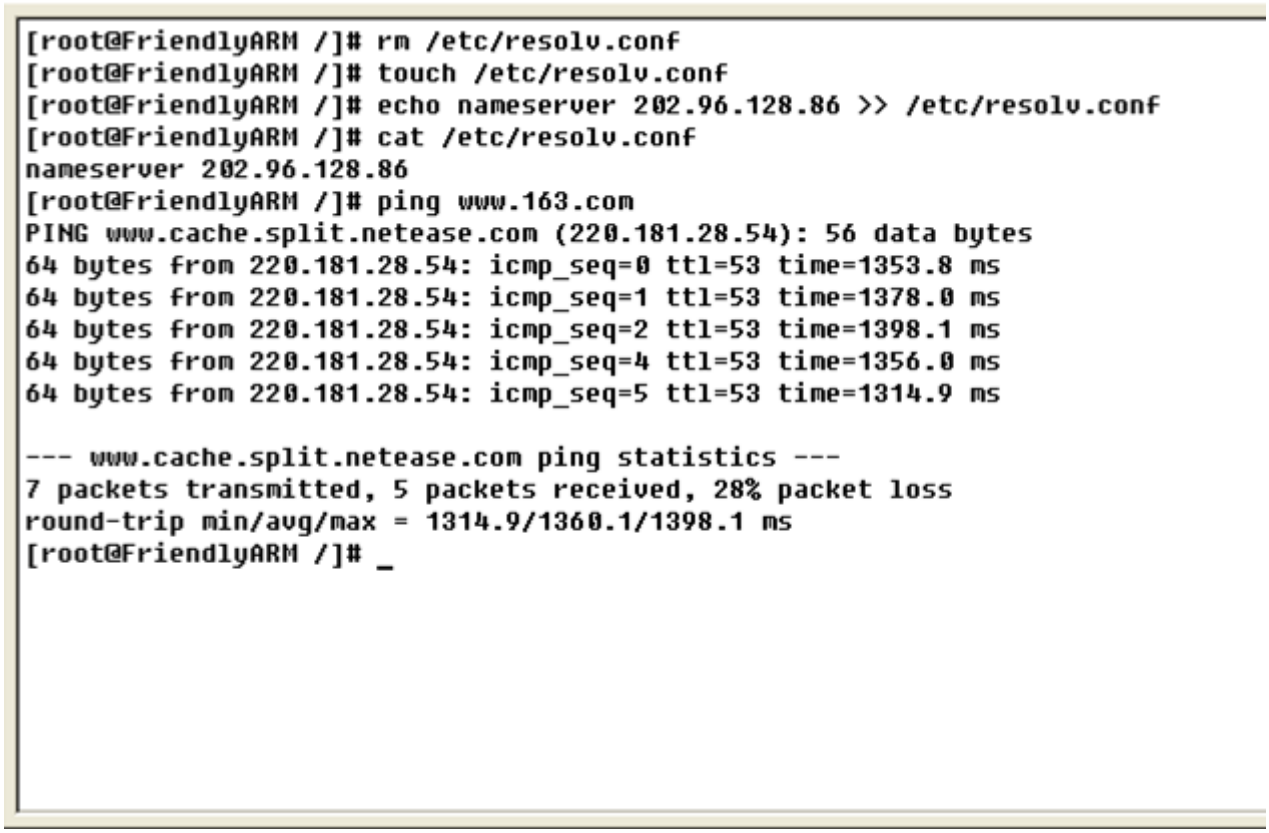


The one in our example was “202.96.128.86”. Therefore we set our board as follows:

#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.

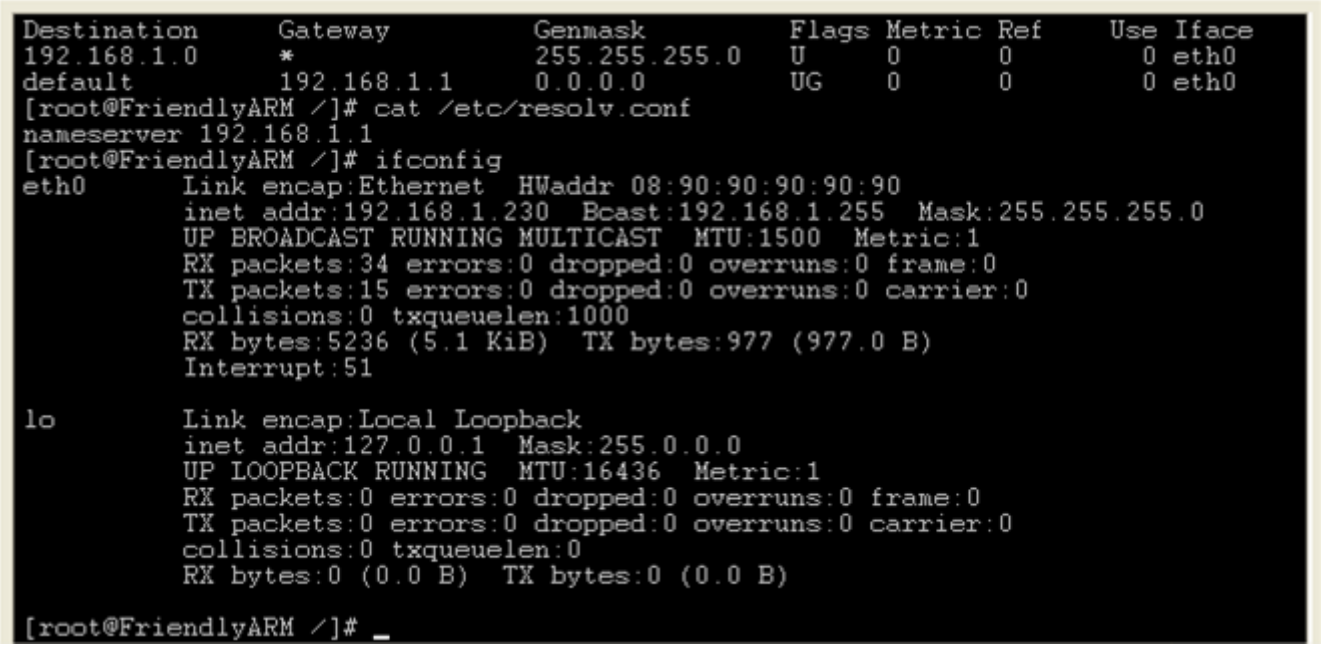


2.19 Configure MAC Address

The MAC address in the Mini6410 is “soft” therefore you can change it via “ifconfig”.

First check your current MAC address via “ifconfig”:

#ifconfig ;



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

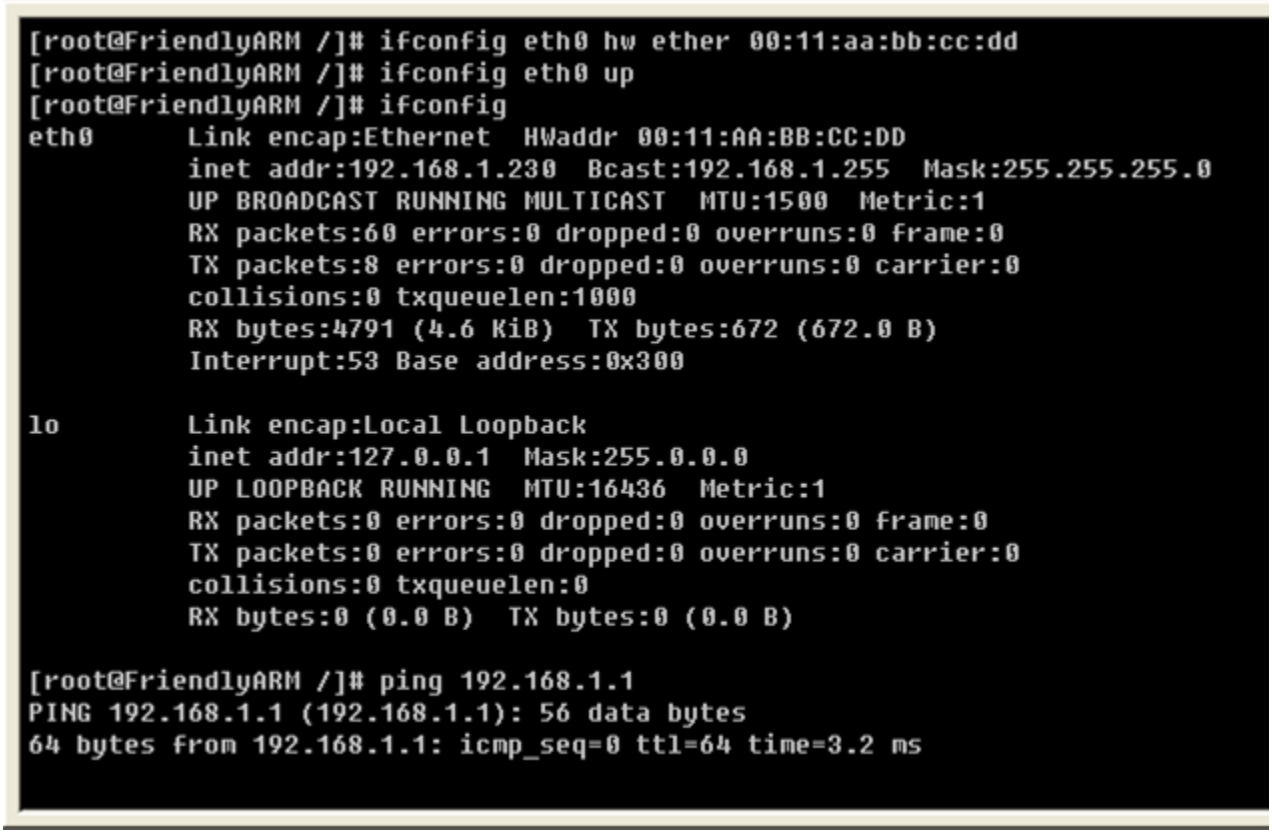
#ifconfig eth0 hw ether 00:11:AA:BB:CC:DD; note: **a,b,c,d,e,f…** could be lower case

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

#ifconfig eth0 up

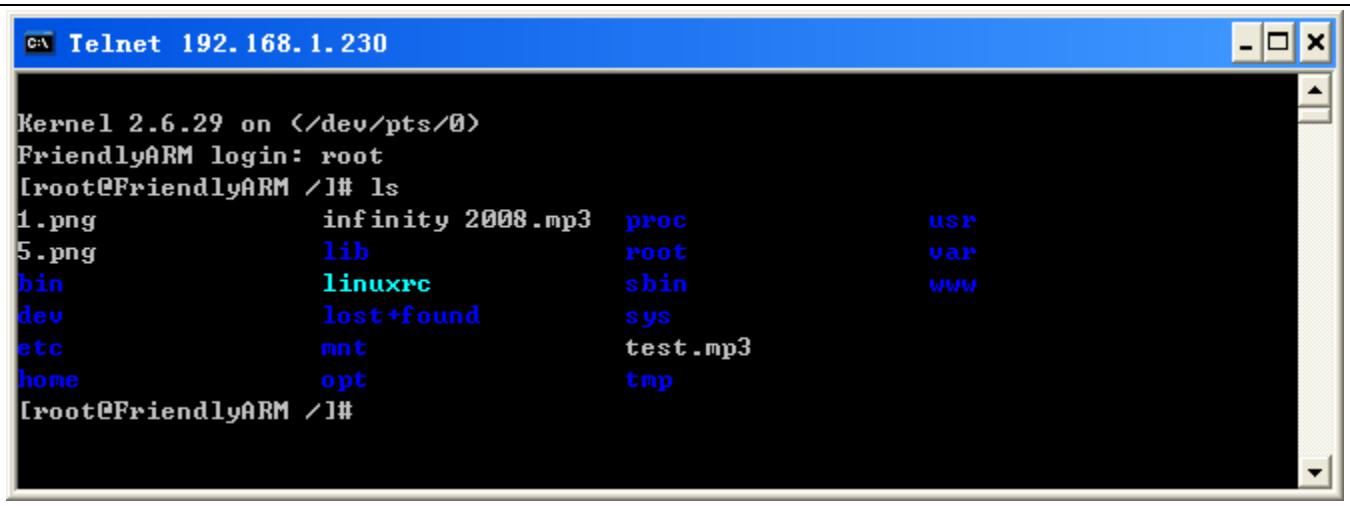
#ifconfig

#ping 192.168.1.1



2.20 Telnet Mini6410

If the system reboots normally it will automatically start a telnet service therefore users can telnet the board too. You can try typing "**telnet 192.168.1.230**" from a command line, type "root" and you will be able to login.

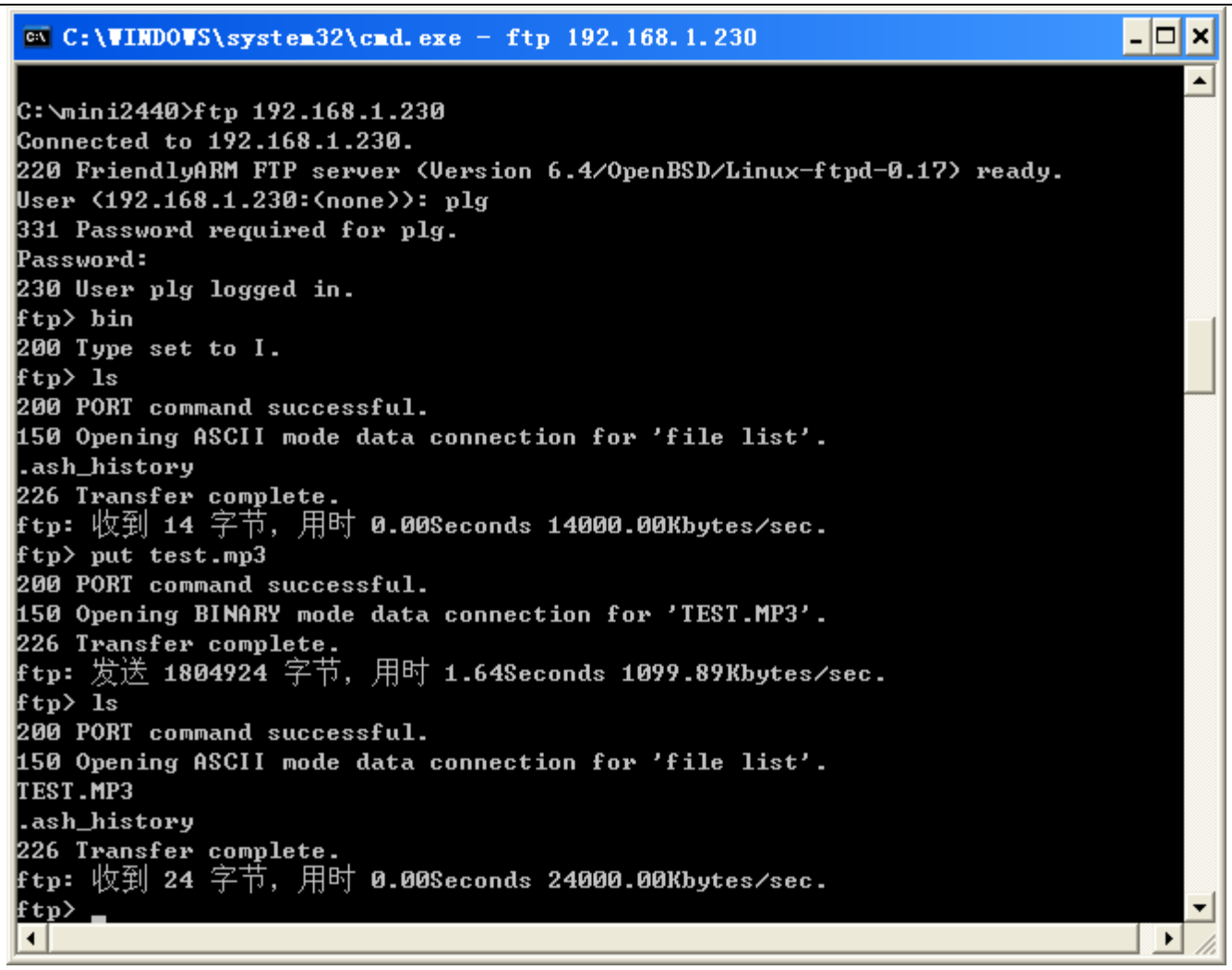


2.21 File Transfer with FTP

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.



2.22 Manipulate LEDs via HTML

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” will manipulate the LEDs via CGI programs 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

2.23 Mount NFS

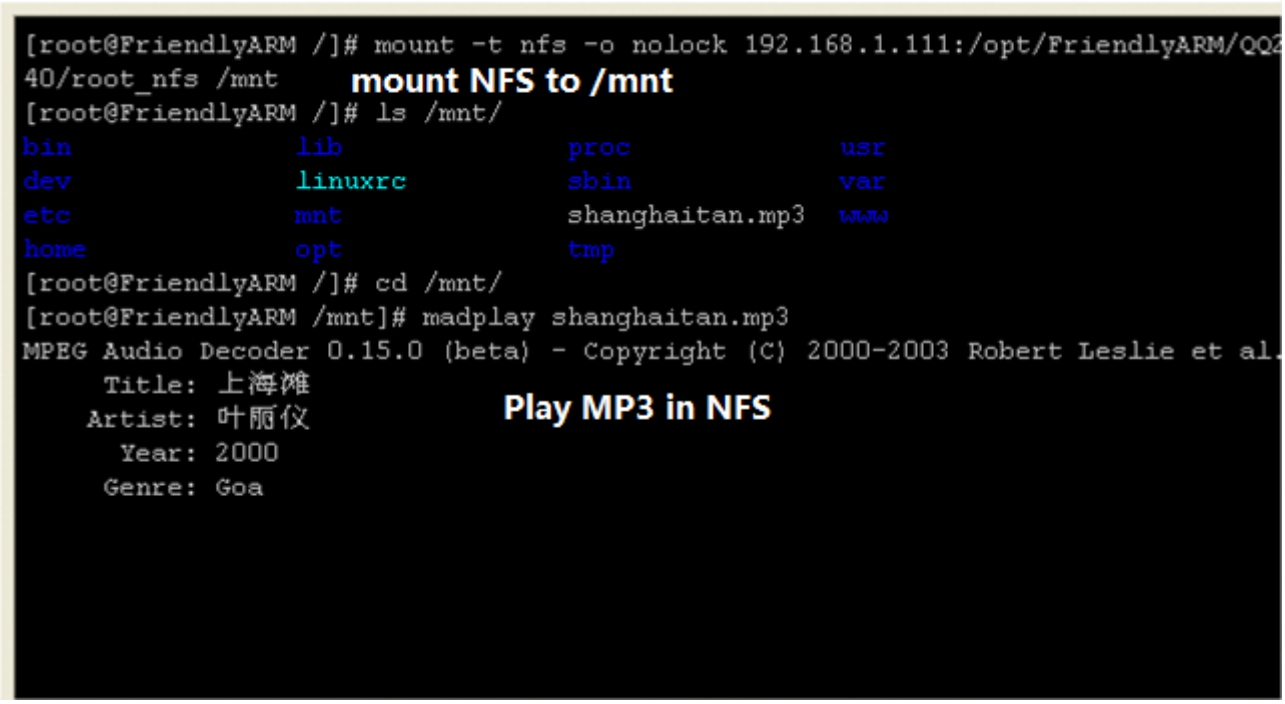
Please make sure you have set up the NFS server in your host PC and then type the following command (our server's IP is 192.168.1.111).

#mount –t nfs –o nolock 192.168.1.111 :/opt/FriendlyARM/mini6410/linux /rootfs_qtopia_qt4 /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**



2.24 Configure System Clock

The Linux command for updating time is "**date**", to synchronize the S3C6420 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 S3C6410'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.

2.25 Save Data to Flash Permanently

The Mini6410 system applies yaffs2 thus can save data dynamically and will not lose any 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.

2.26 Set up Auto Run Programs 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
```

2.27 Take Screenshots with Snapshot

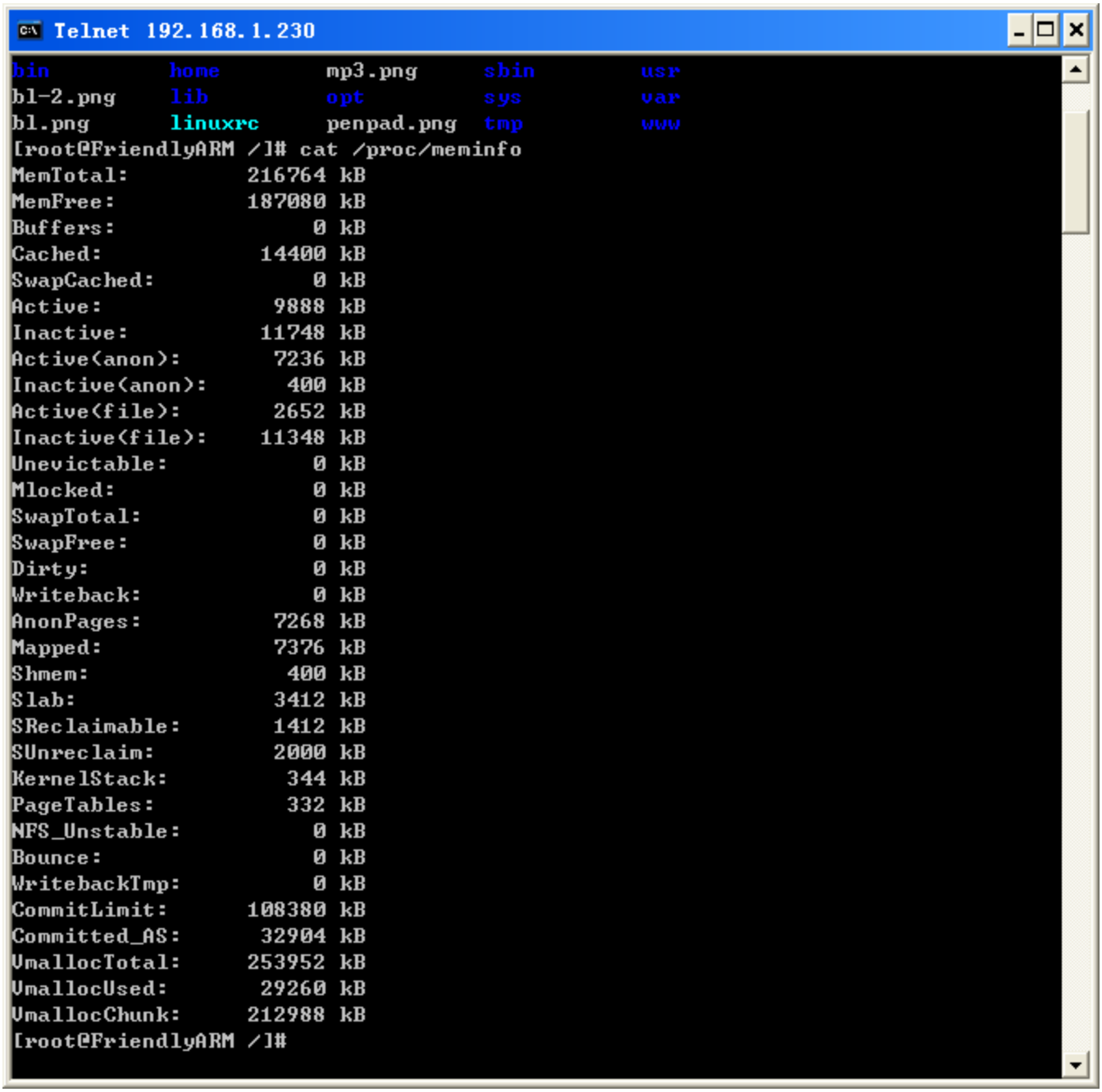
Users can take screenshots with “snapshot” and save them as png files

#snapshot pic.png

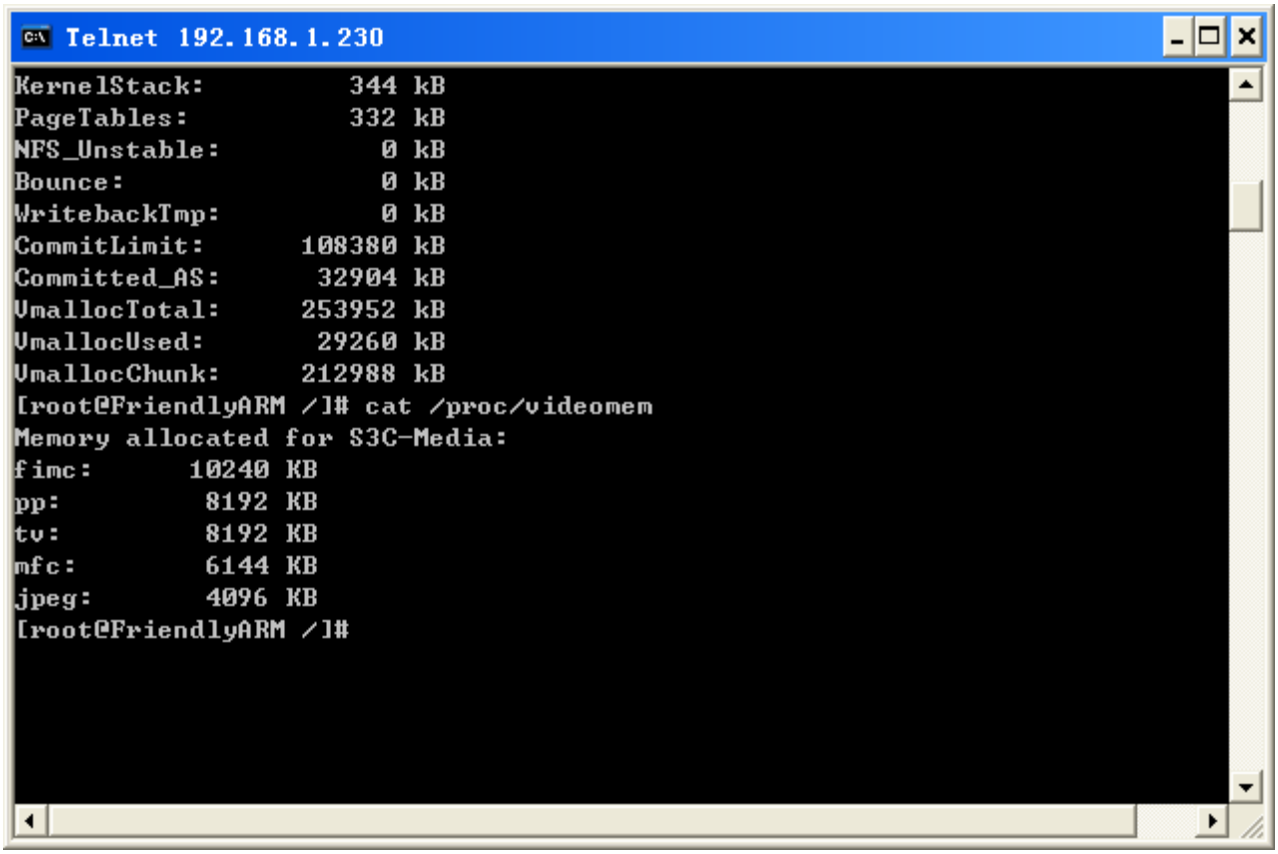
Executing this command will take a screenshot of the current LCD display and save it as “pic.png”.

2.28 Check RAM Info

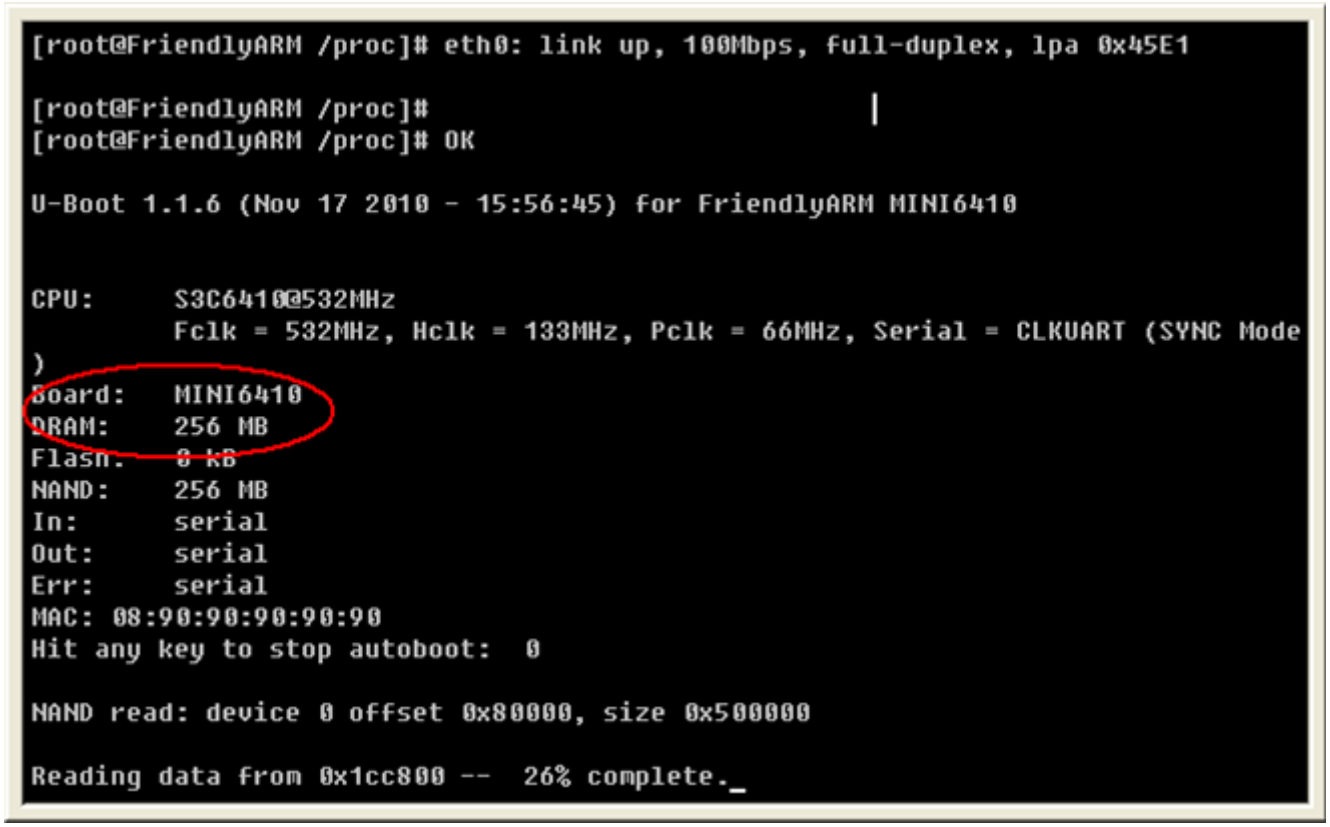
The Mini6410 system incorporates a 256M DDR RAM. Some users complained that they can only find 68M available this is because the multi-media driver takes quite a lot. In general users can check the RAM info by commanding “cat /proc/meminfo”, however this only shows the amount available to the system. The total is 216M



Actually the 6410 system's multi-media co-processor takes some RAM too. Users can get more details by commanding "cat /prov/videomem".



Before the system is booted you can check the actual RAM too.


[root@FriendlyARM /proc]# eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
[root@FriendlyARM /proc]#
[root@FriendlyARM /proc]# OK
U-Boot 1.1.6 (Nov 17 2010 - 15:56:45) for FriendlyARM MINI6410
CPU: S3C6410@532MHz
Fclk = 532MHz, Hclk = 133MHz, Pclk = 66MHz, Serial = CLKUART (SYNC Mode
)
Board: MINI6410
DRAM: 256 MB
Flash: 0 kB
NAND: 256 MB
In: serial
Out: serial
Err: serial
MAC: 08:90:90:90:90:90
Hit any key to stop autoboot: 0
NAND read: device 0 offset 0x80000, size 0x500000
Reading data from 0x1cc800 -- 26% complete.

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 Mini6410 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 it is easy to be installed and set up. Fedora 10 and later versions are more complicated and therefore may not be easy for beginners and Fedora 8 and earlier versions are a little bit obsolete. Please follow the steps below to install.

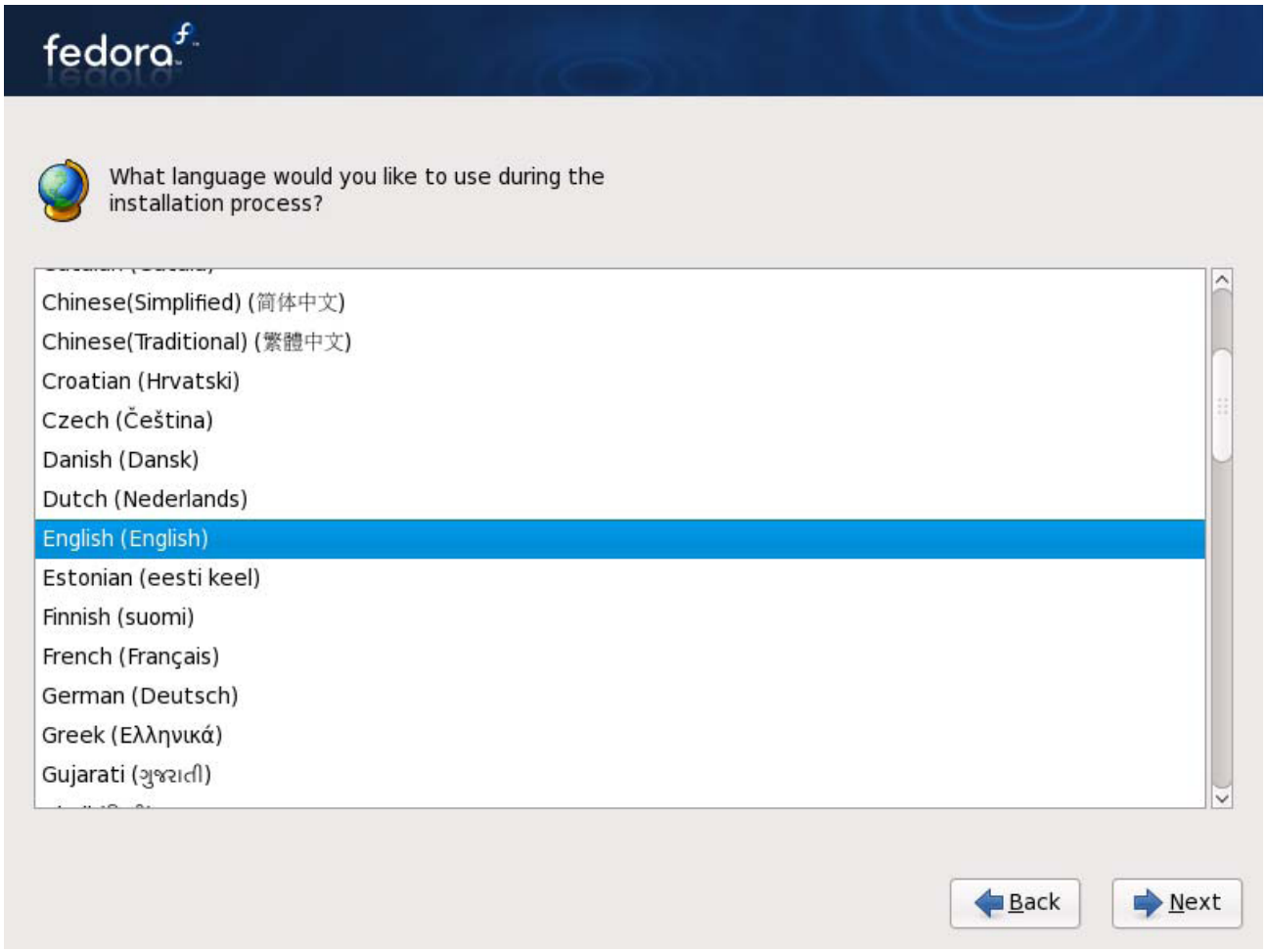
3.1 Install Fedora 9.0

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"

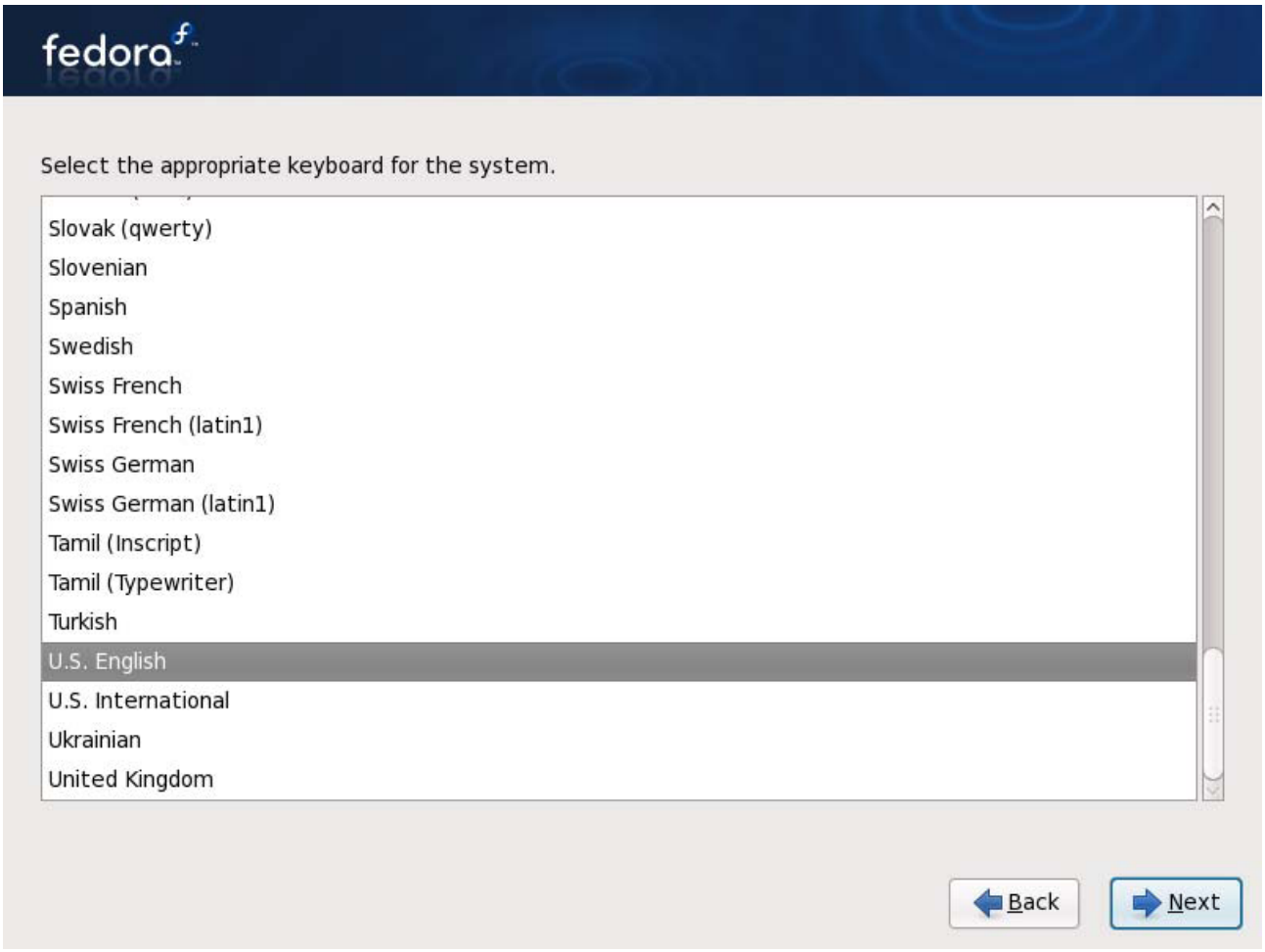
Step2：The system will check the installation disk. It can be ignored, just press “Skip” to the next step

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

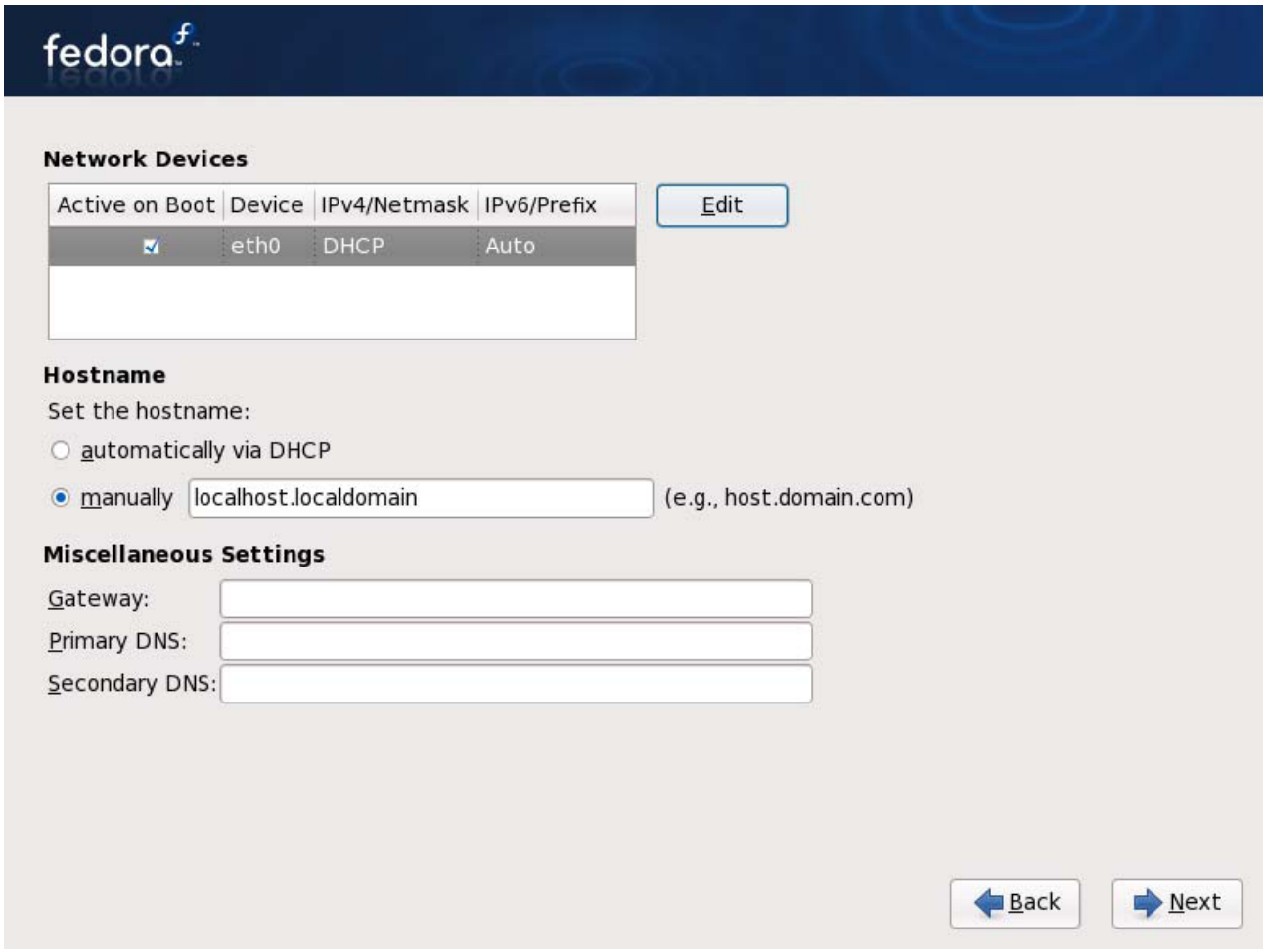
Step4：set the installation language. In this example, we chose the simplified English.



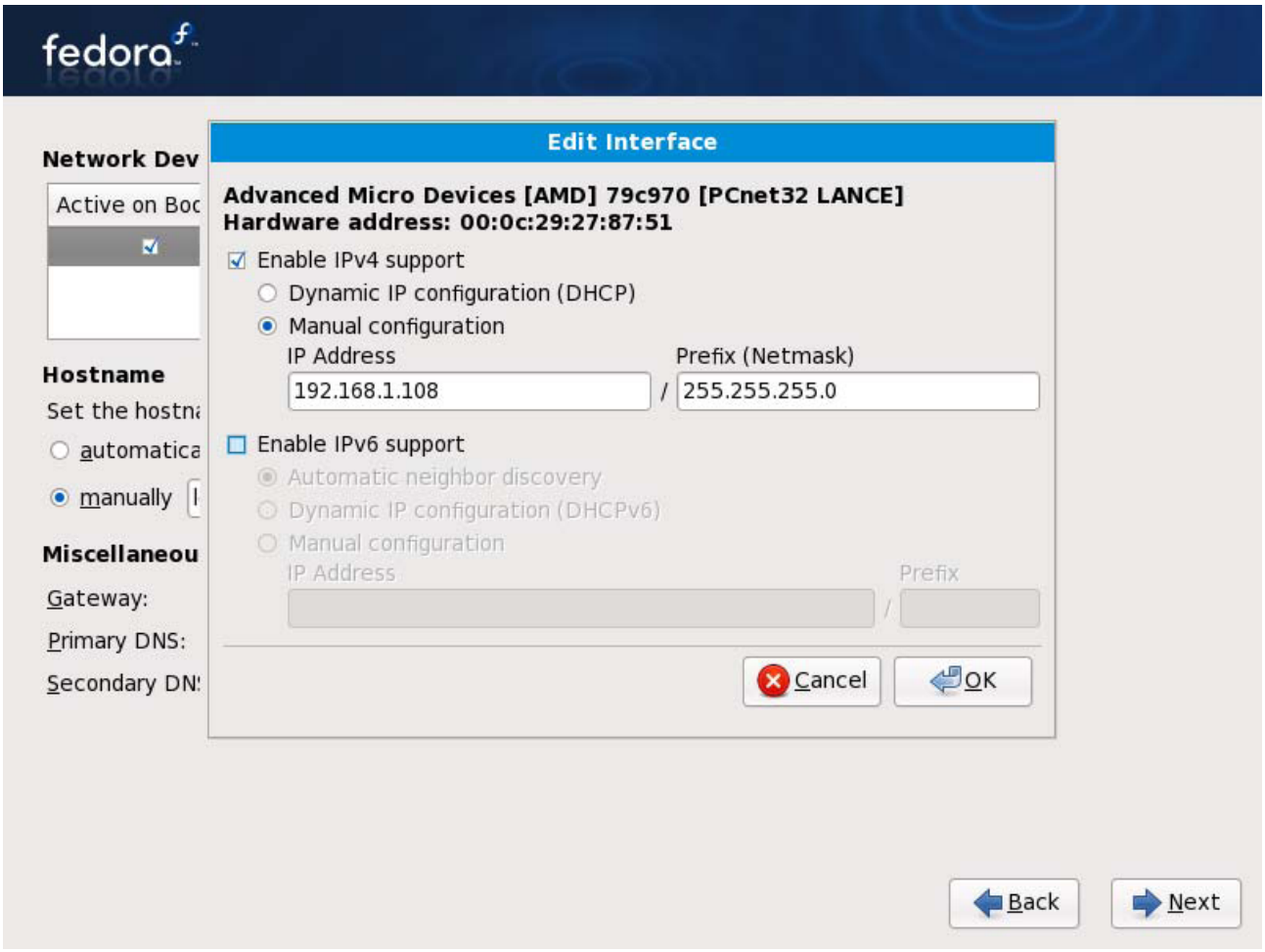
Step5：set the keyboard, in this example, we chose the U.S. key board.



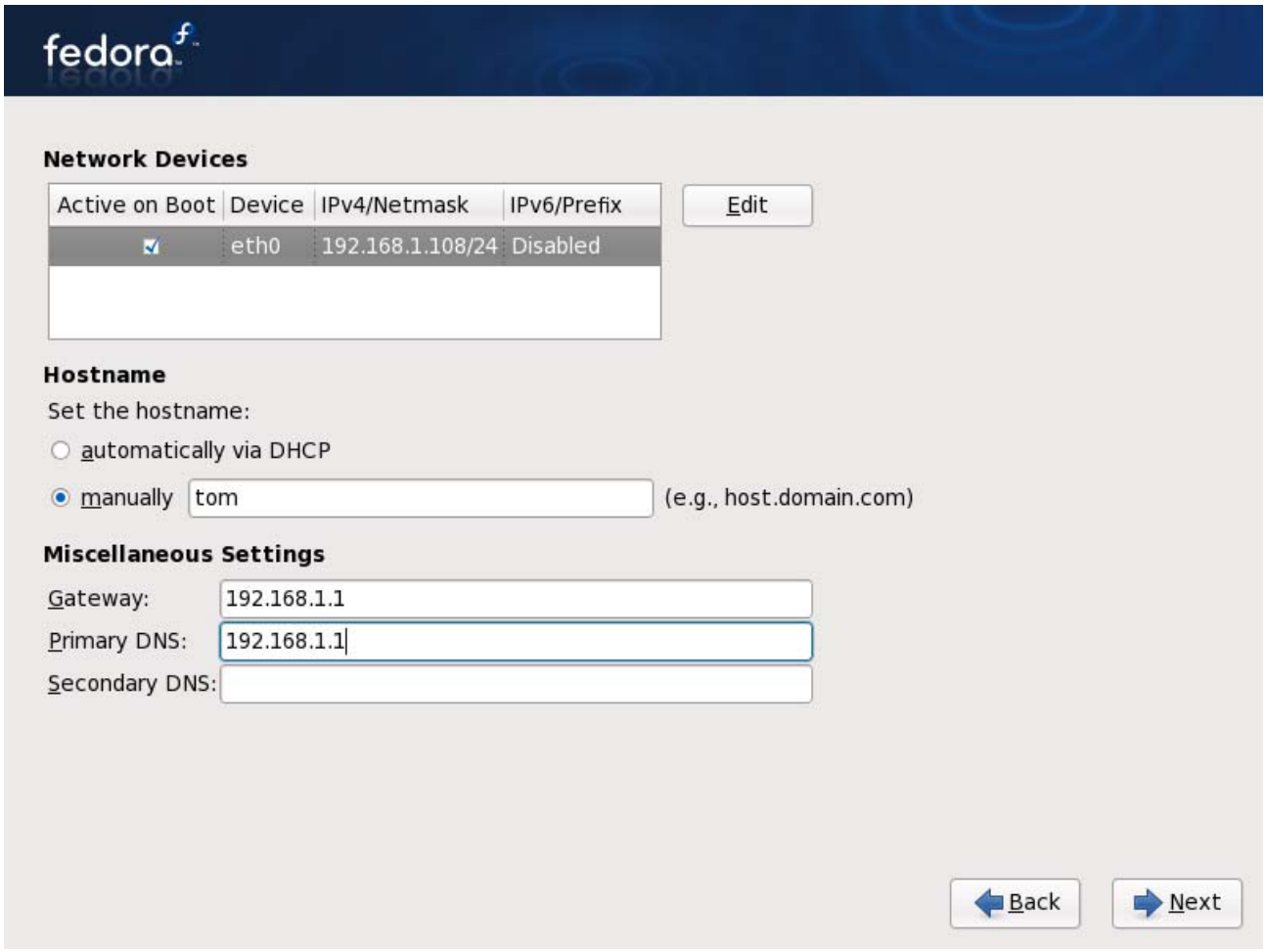
Step 6：configure the network.



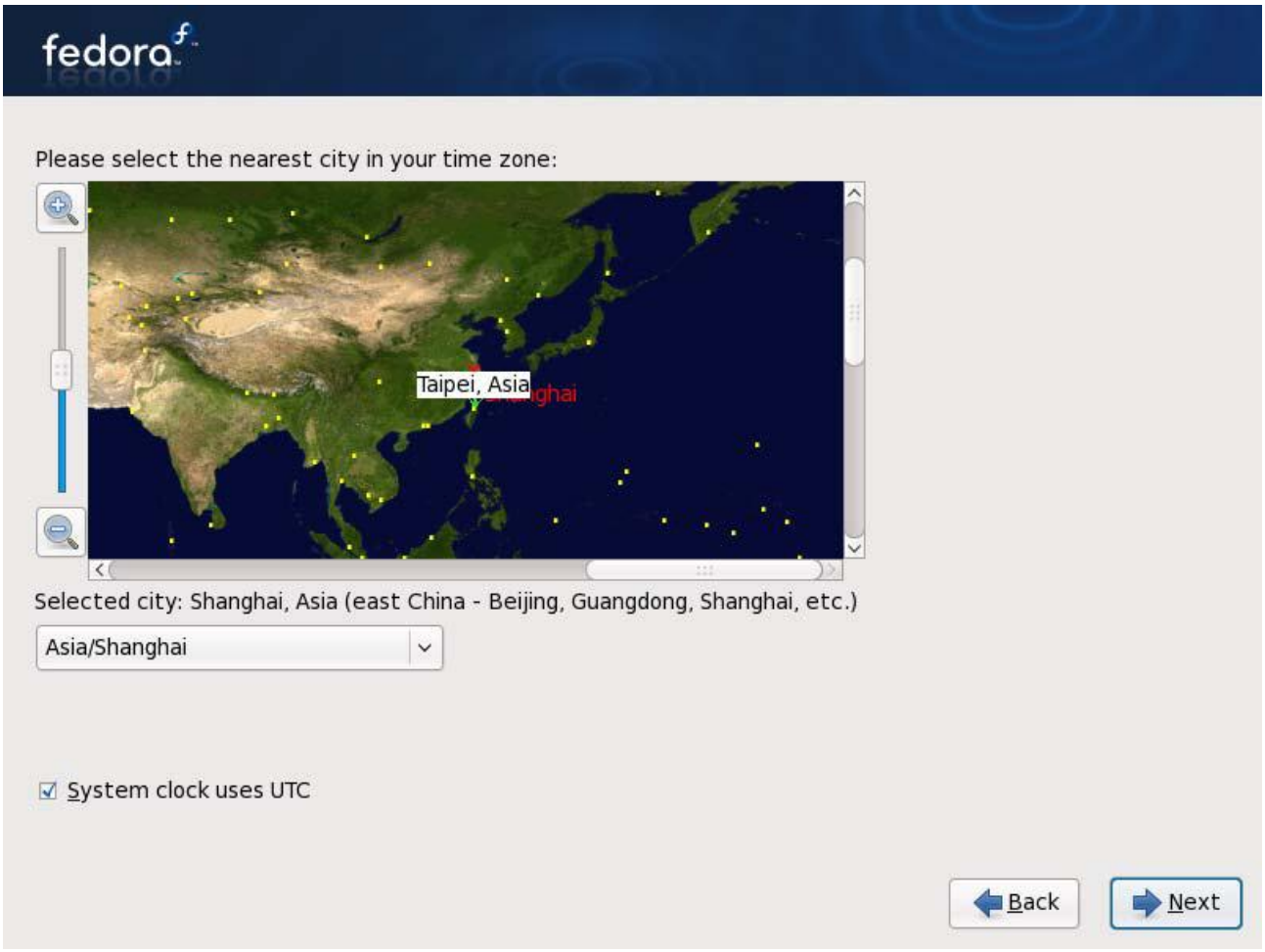
In our example, we didn't set it as "DHCP", we used a static IP instead, and typed the IP and subnet mask as follows.



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

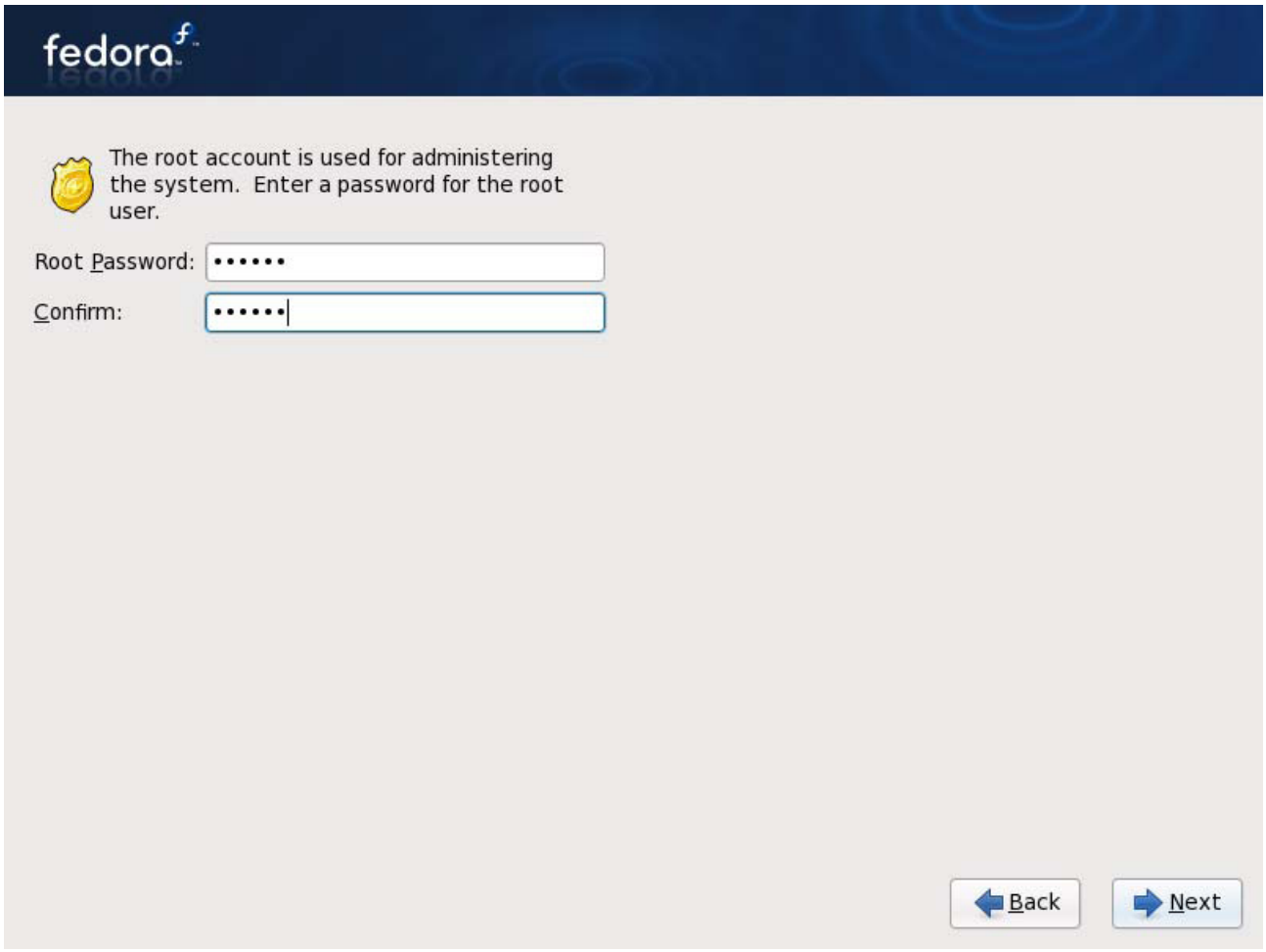


Step 7：set the time zone. We chose “Asia/Shanghai”.

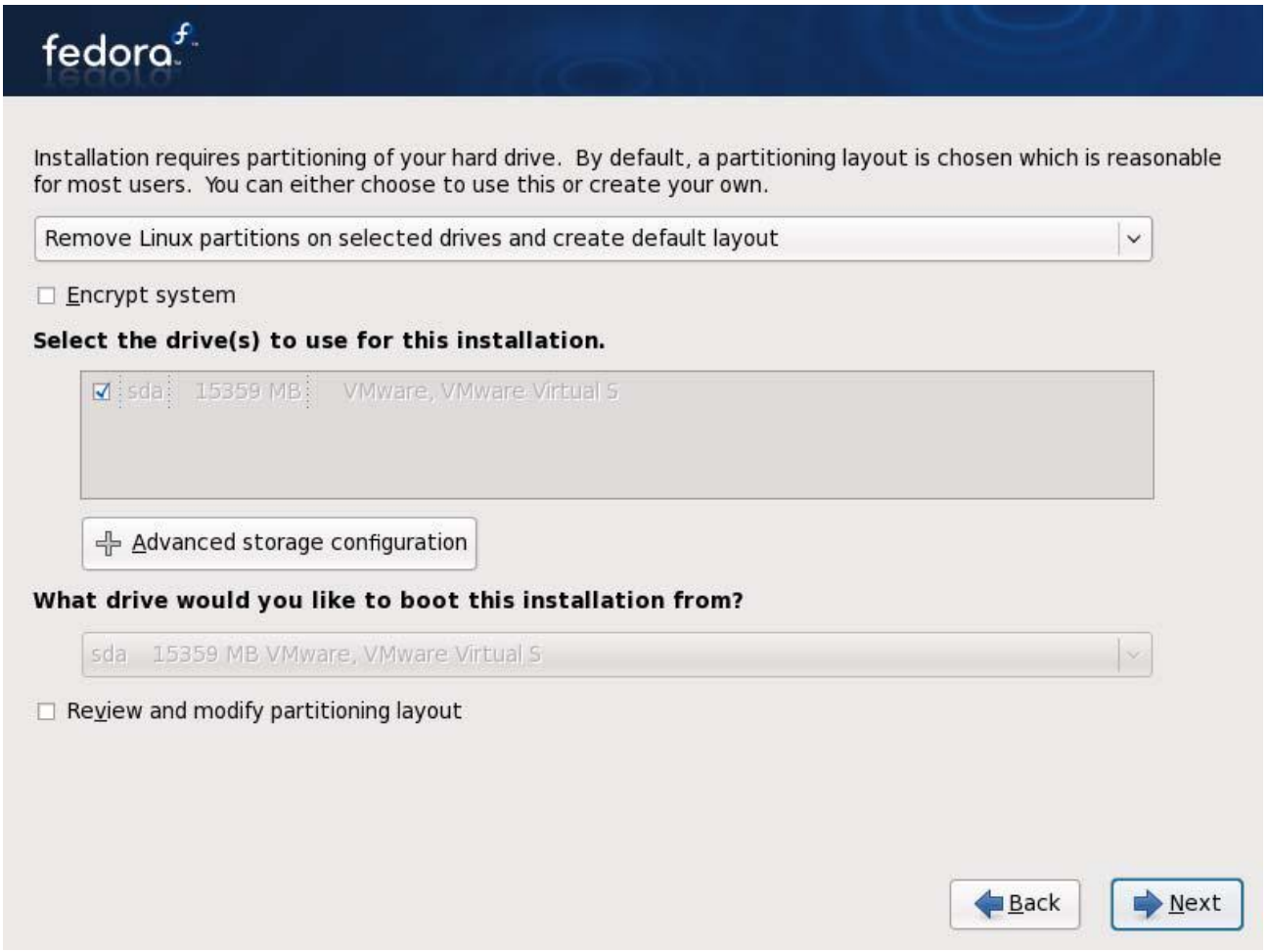


Step 8：set up the administrator's password, i.e. the root's password. "root" is the super user.

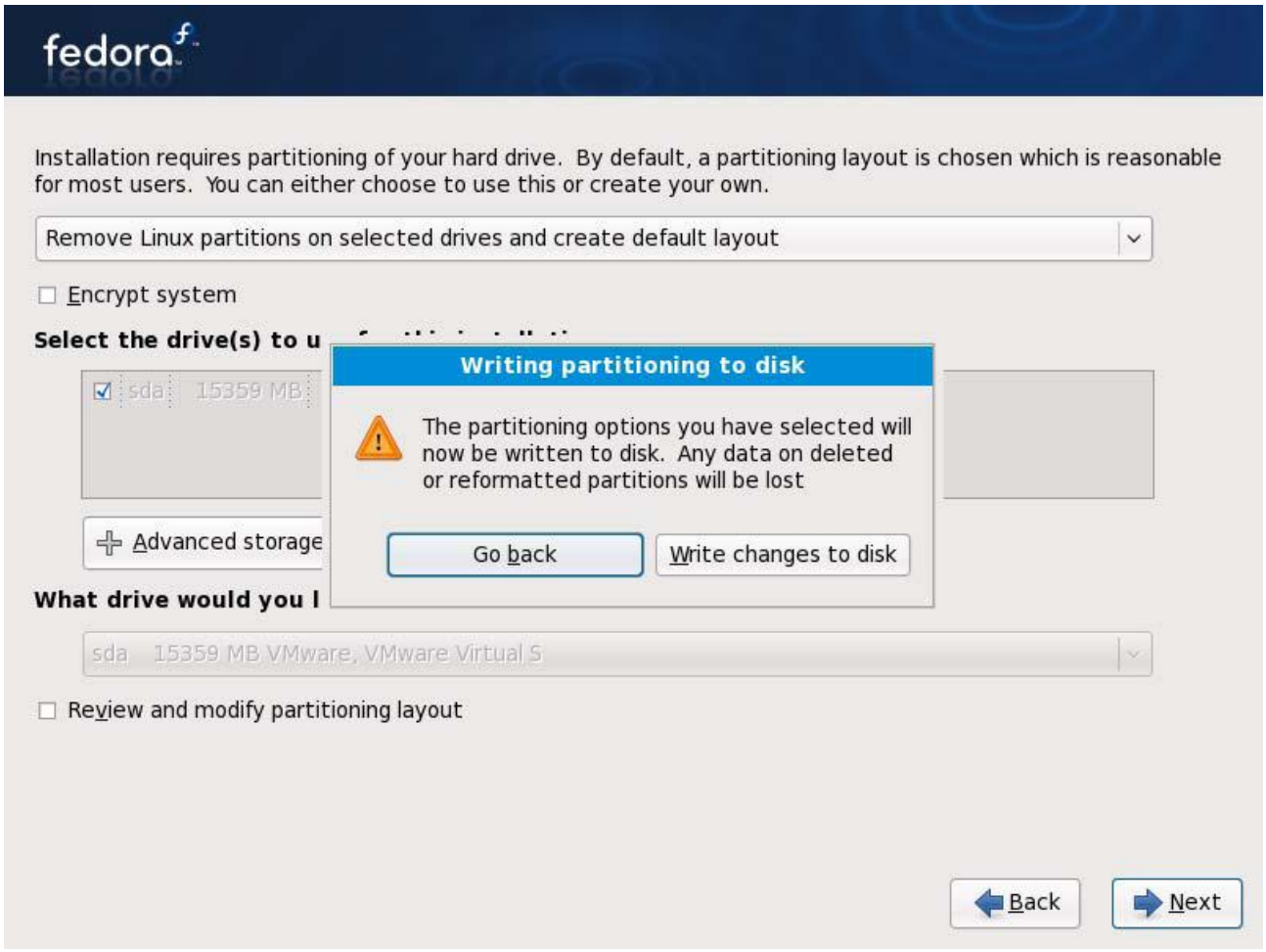
It should be at least 6 characters



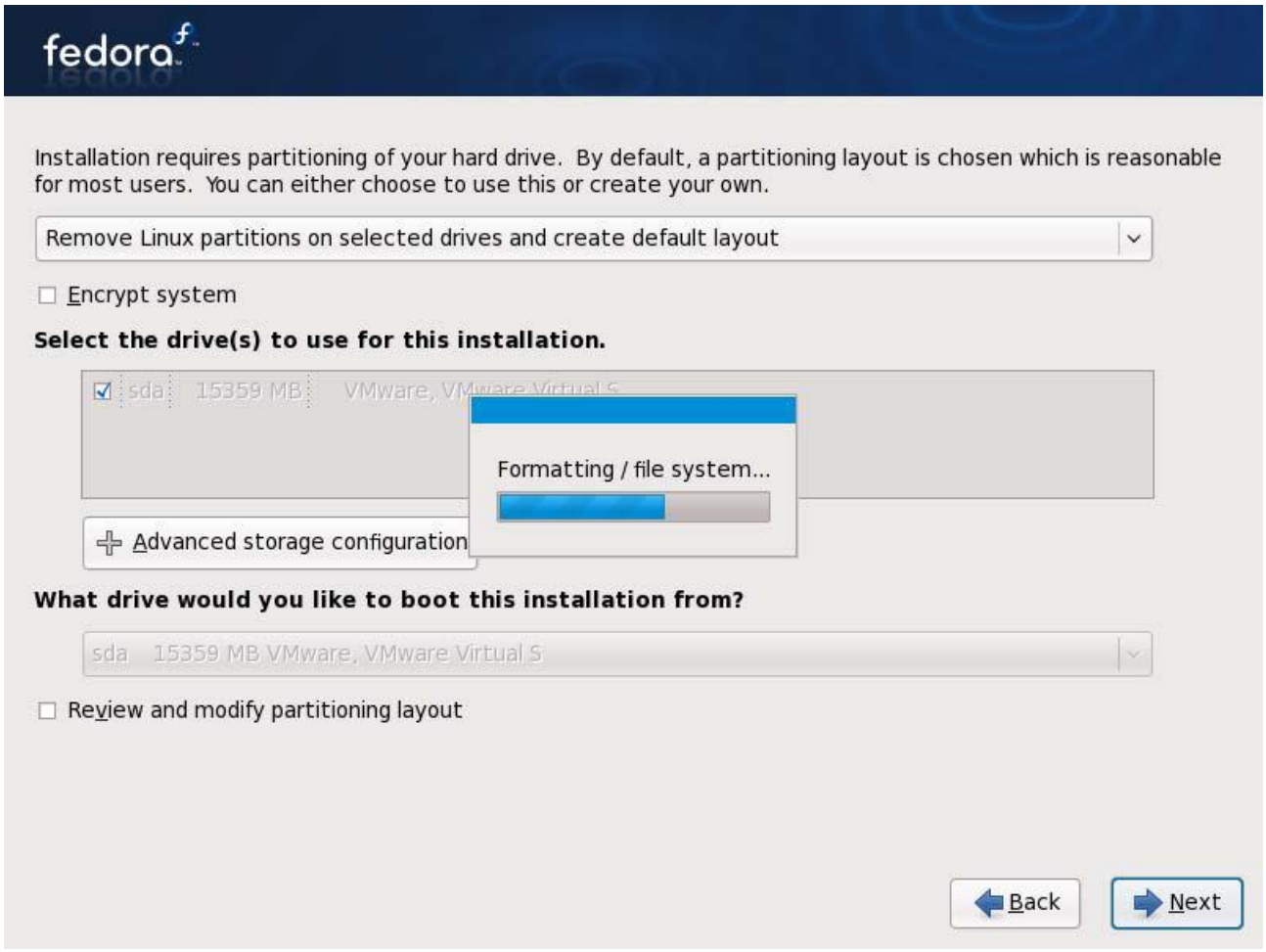
Step 9：disk partition. We followed the default option. Before do this, please back up disk data.



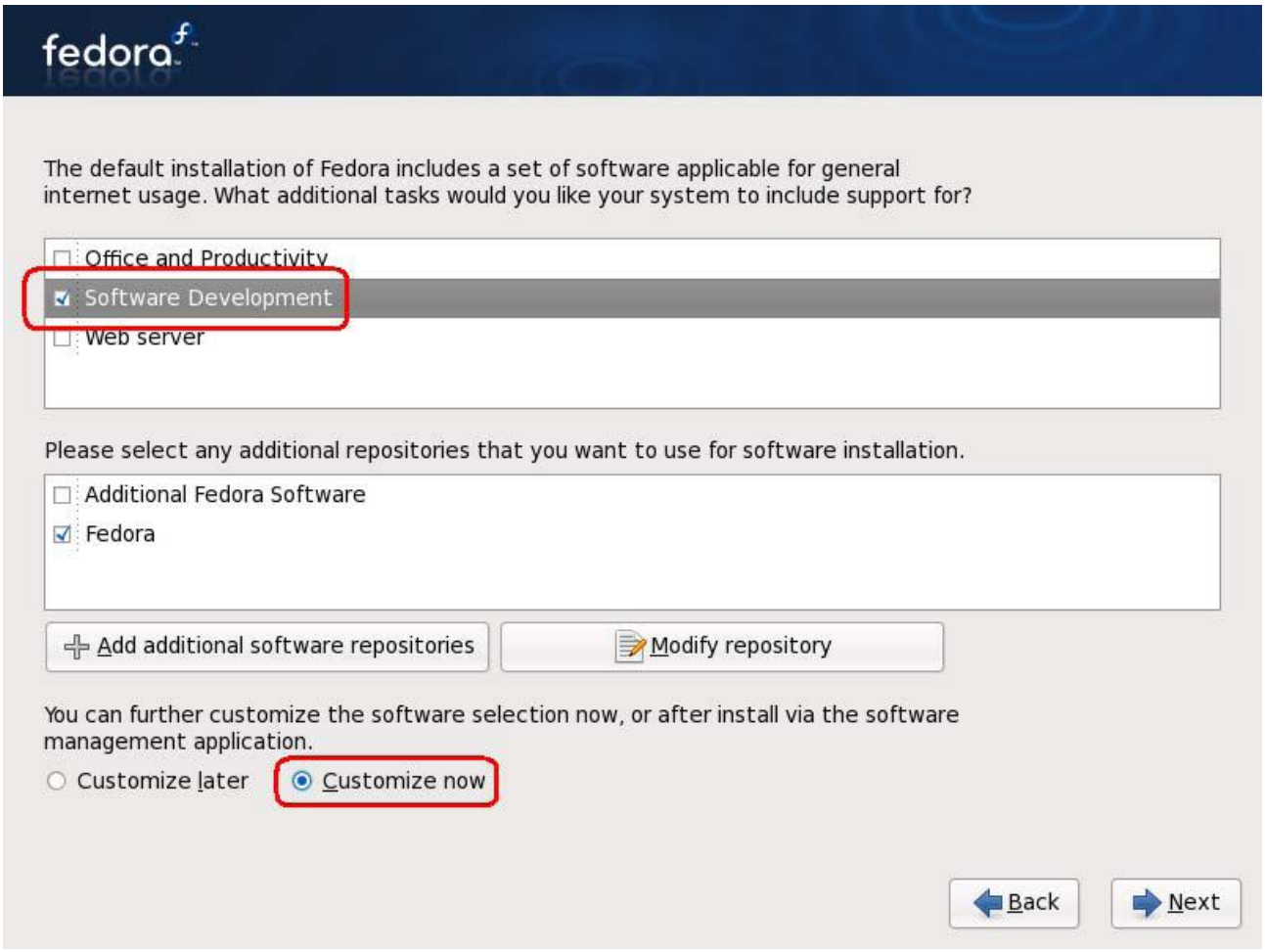
Click on “Next”, it will warn the user that all the data will be deleted. Usually we would do this installation in VMWARE, so we chose “Write changes to disk” and disk format would begin.



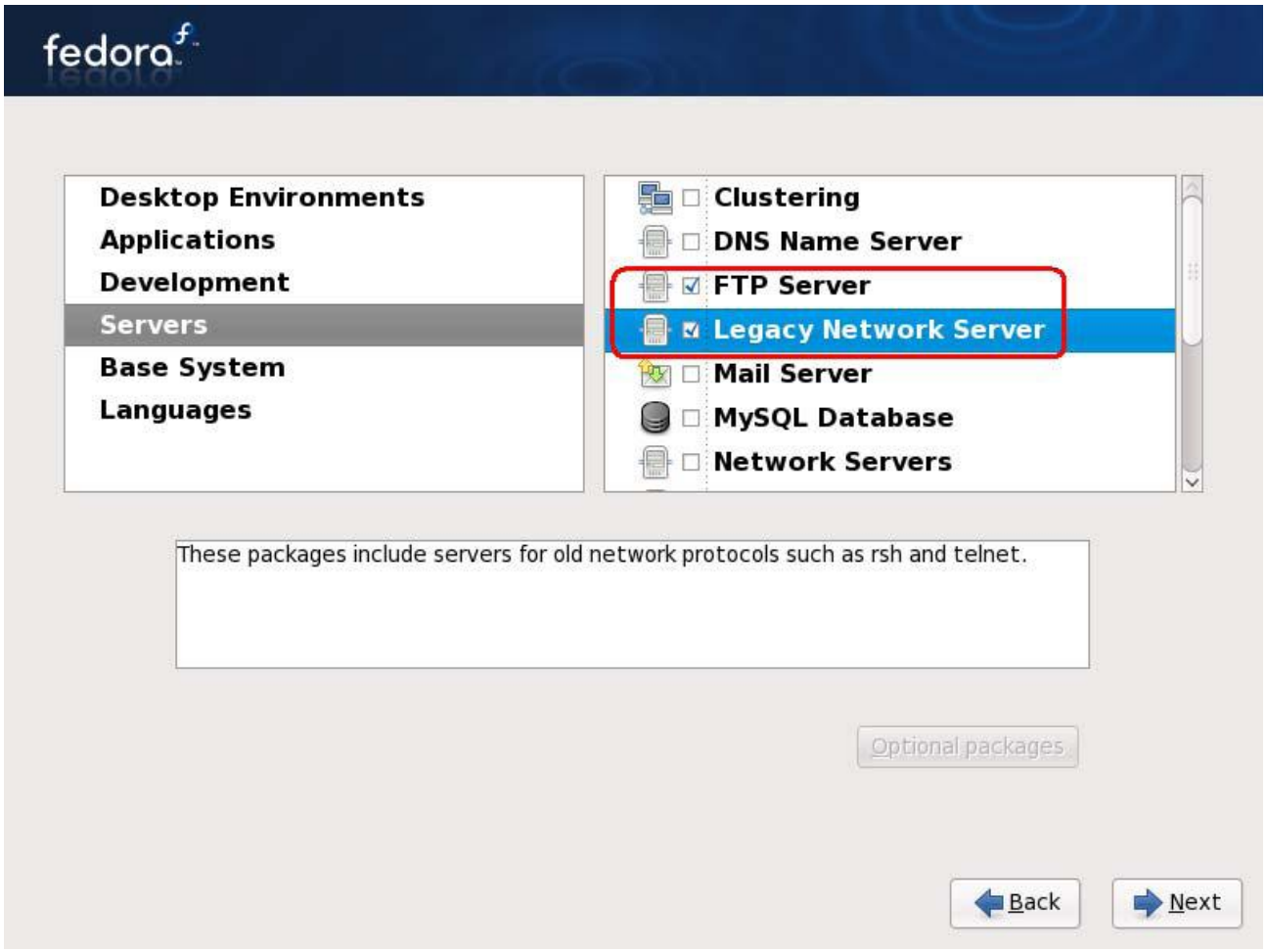
Here is the format process:



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

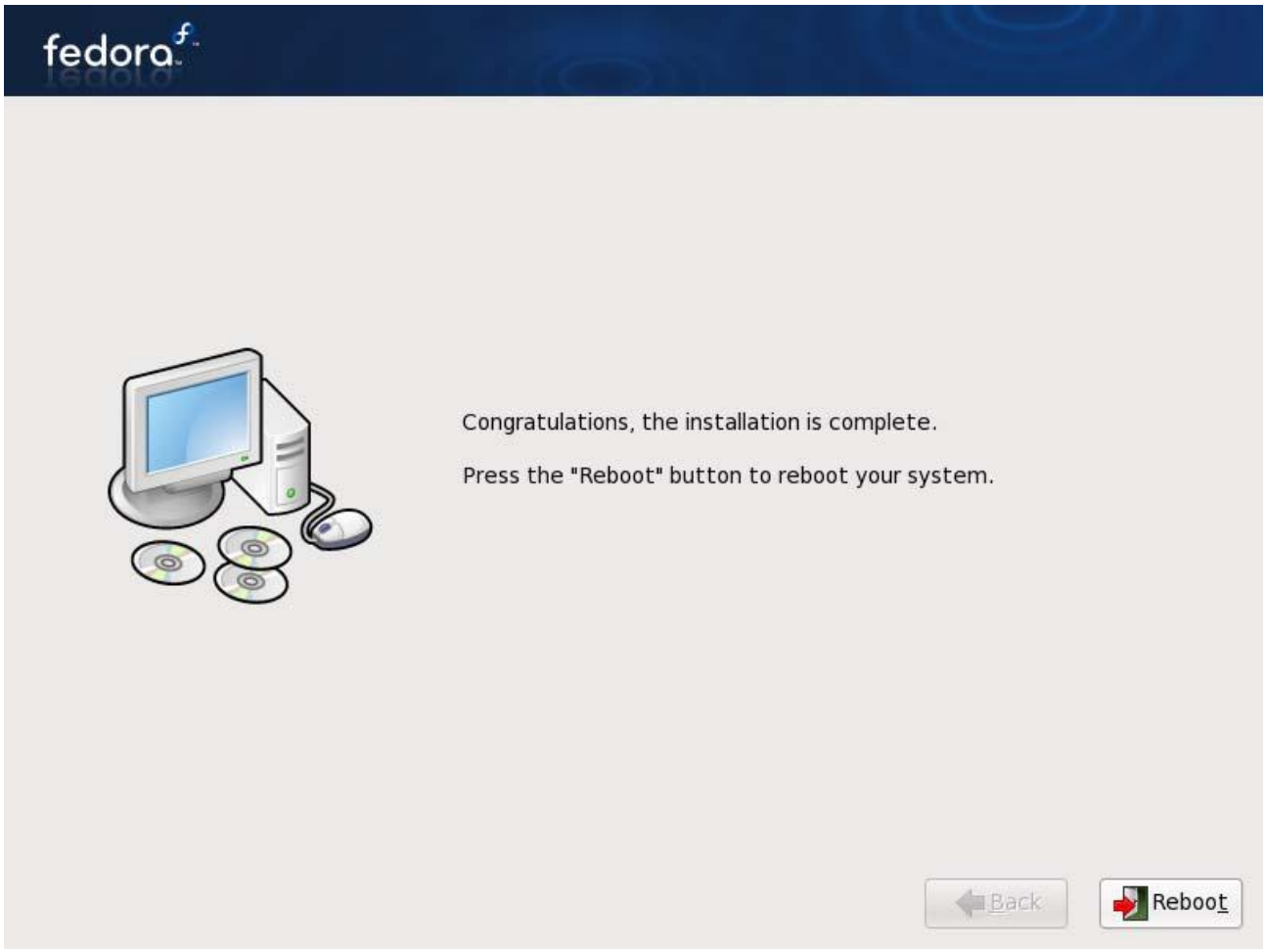


Step 12：configure the “server” item as follows:



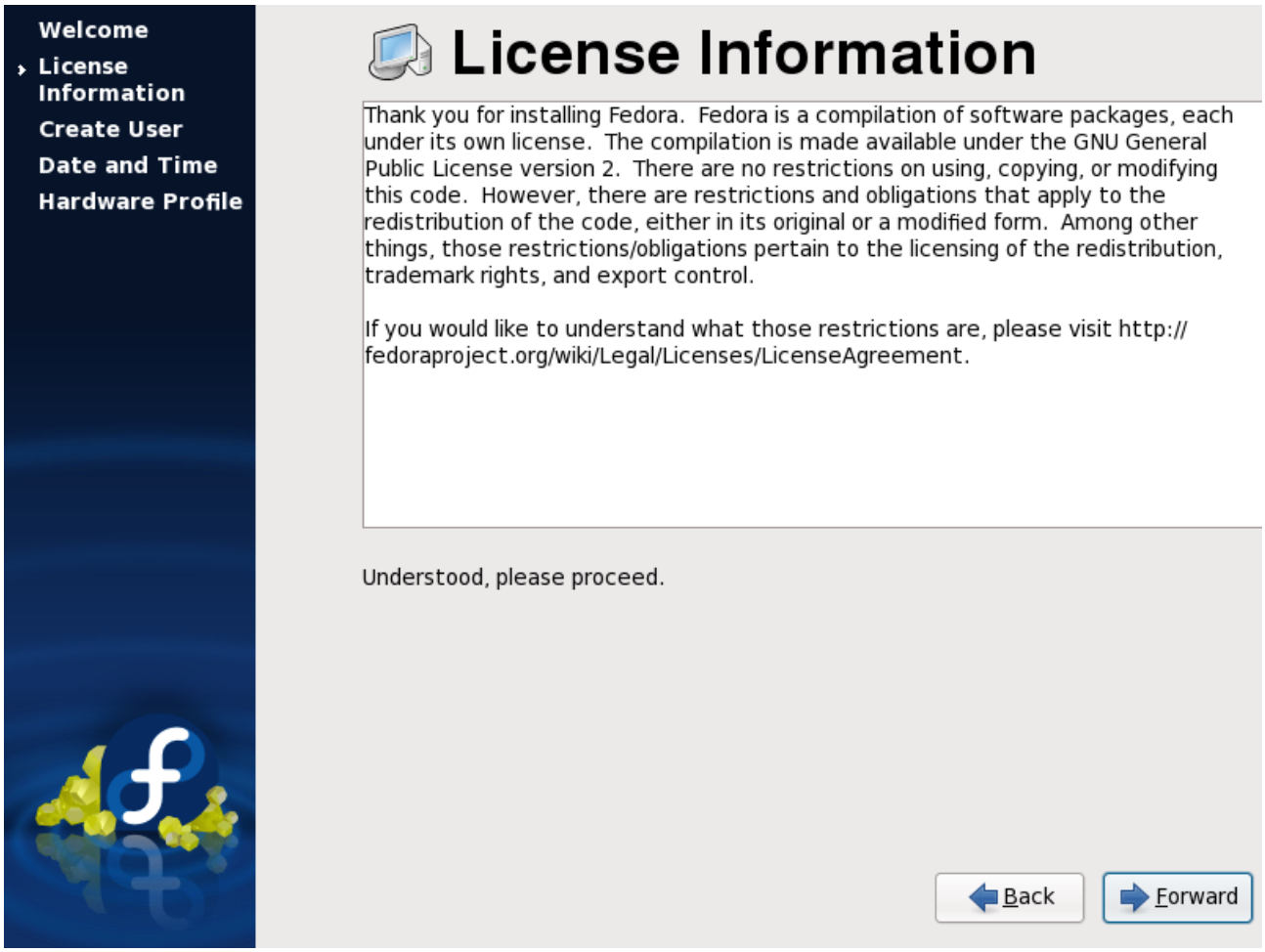
Step 13：begin installation

Step14: installation complete.

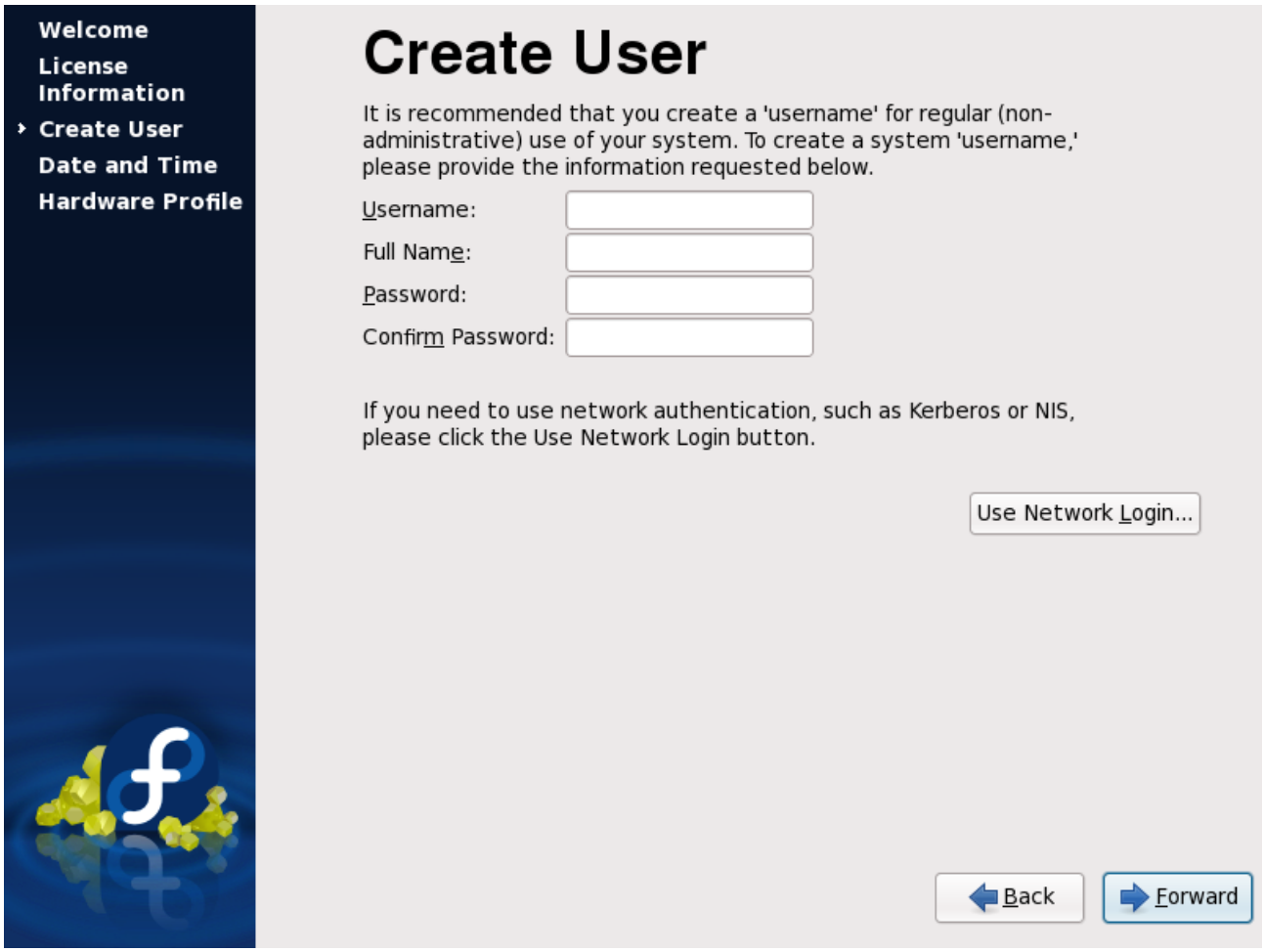


Step15: after installation completed, click on the reboot button on the page shown in step 14

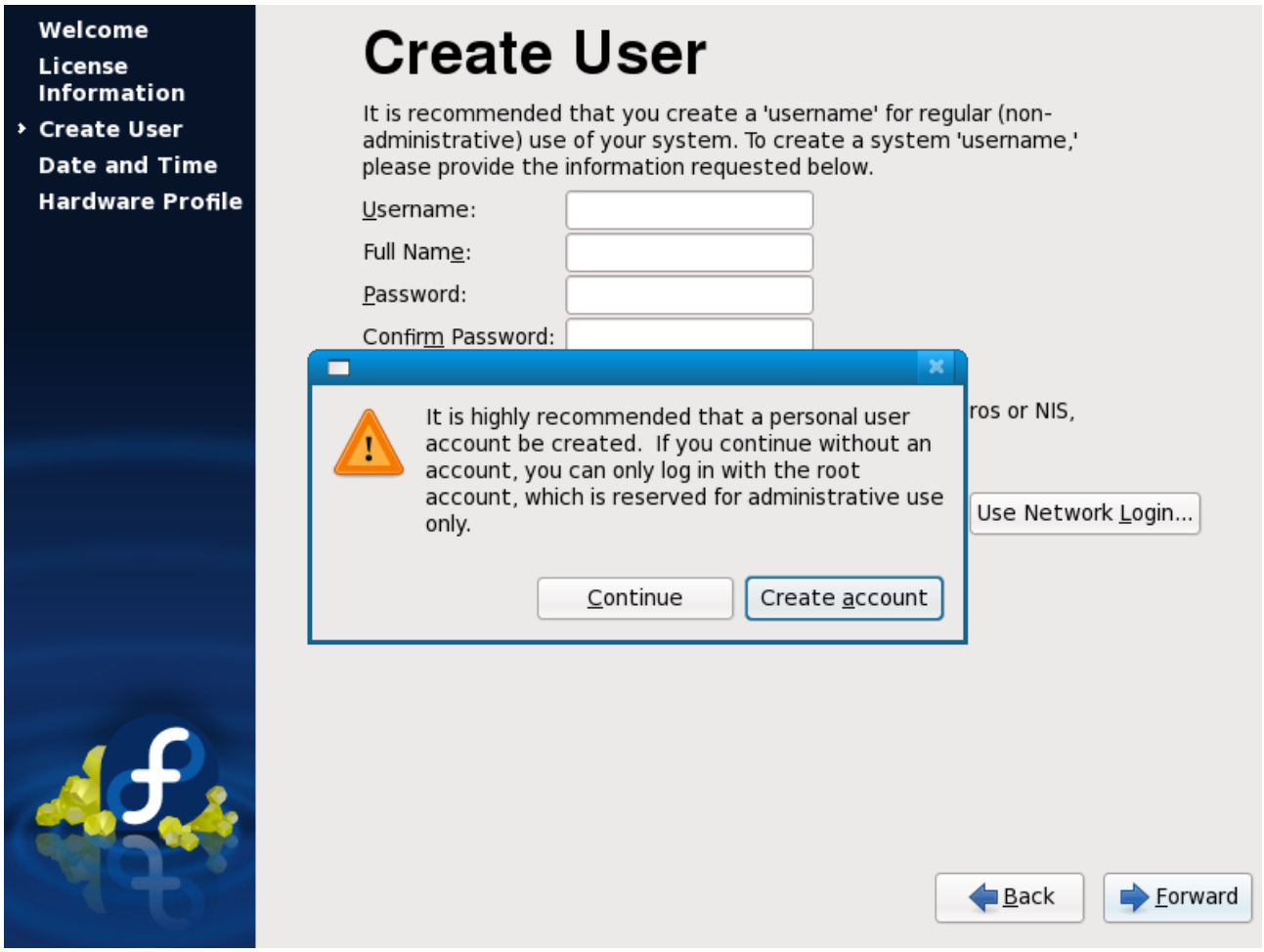
Step16: skip this license page and go “forward”



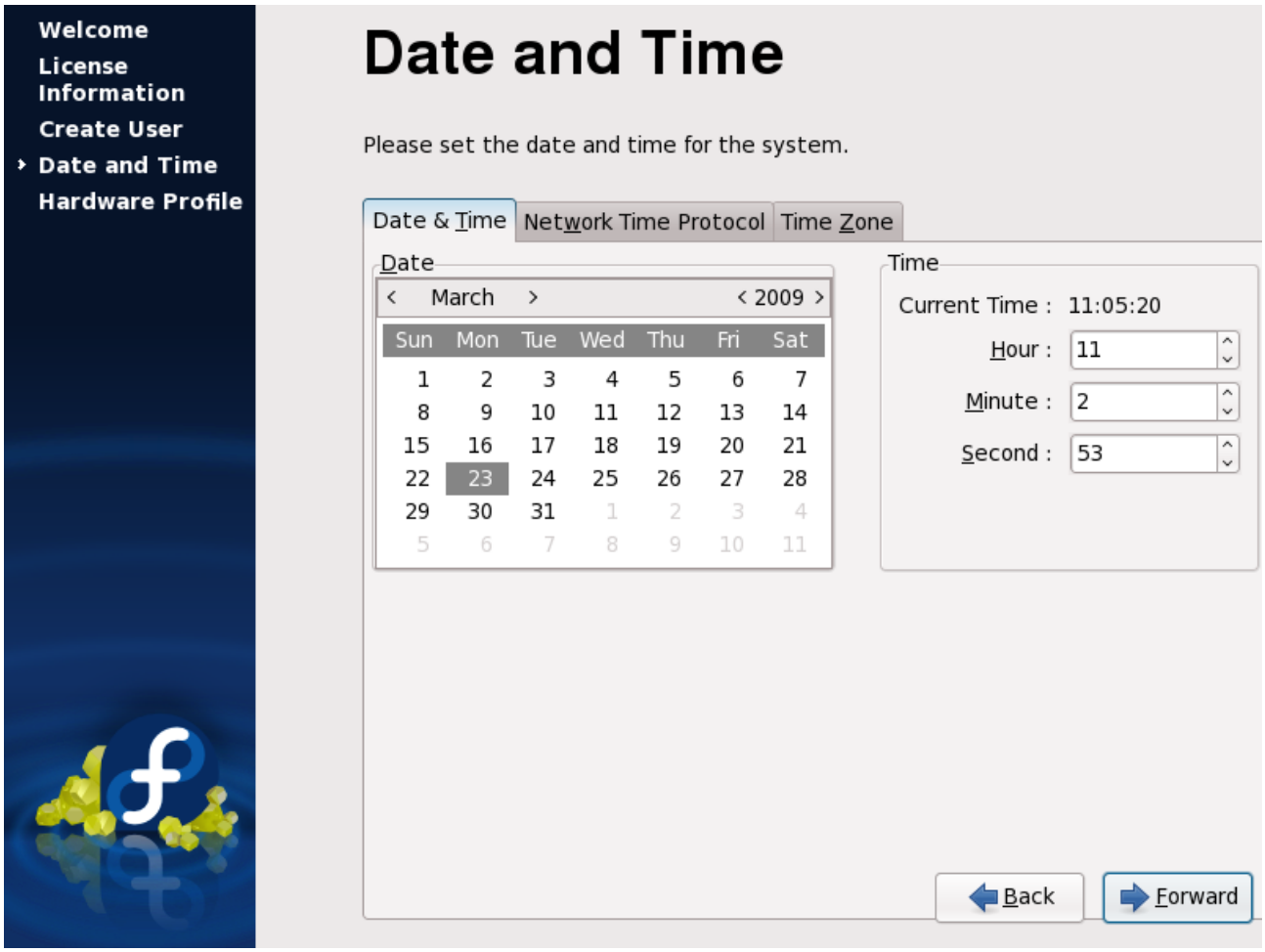
Step17: create new users. We ignored user creation and went to the next step.



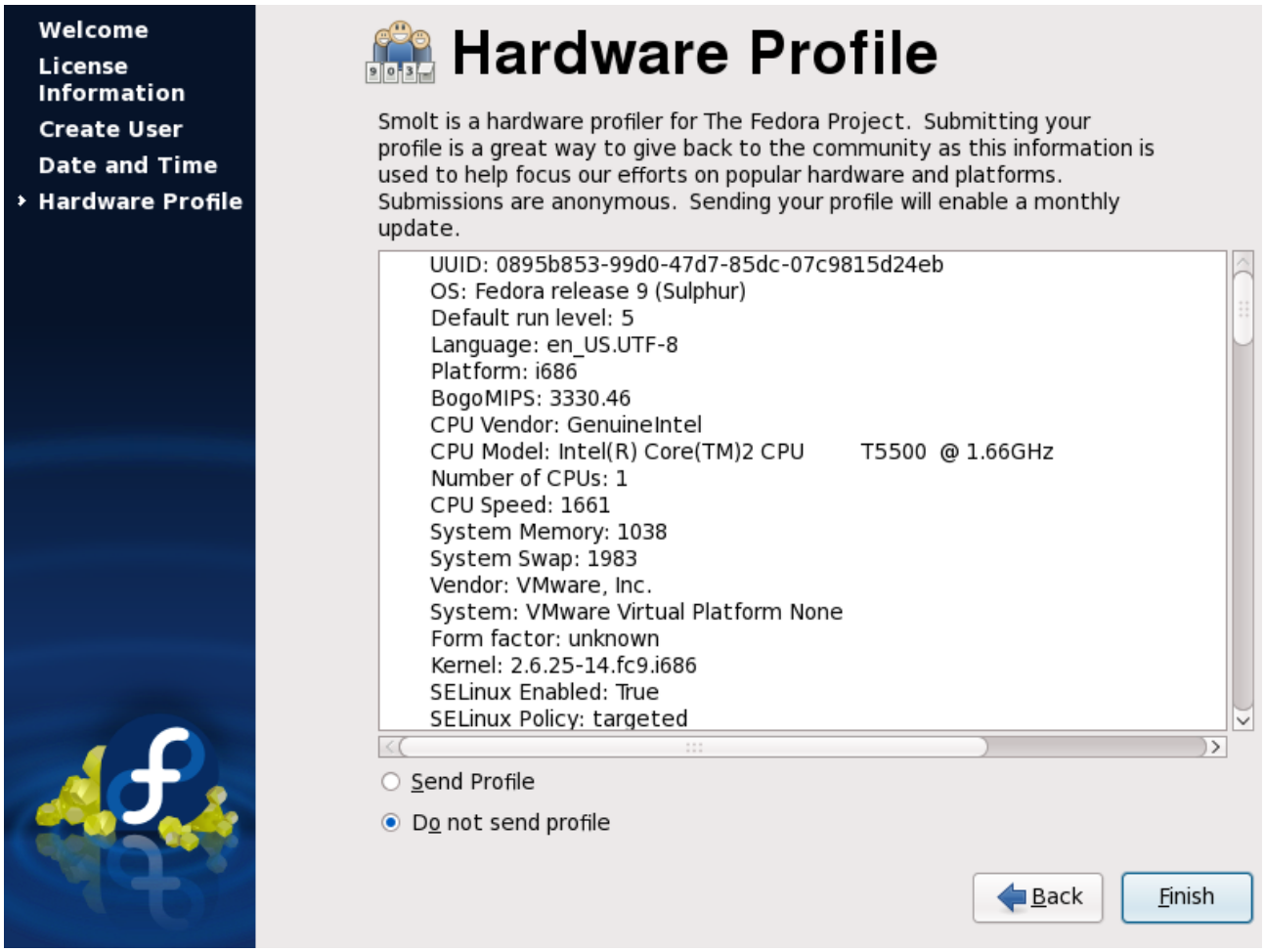
Press “continue” to go on.



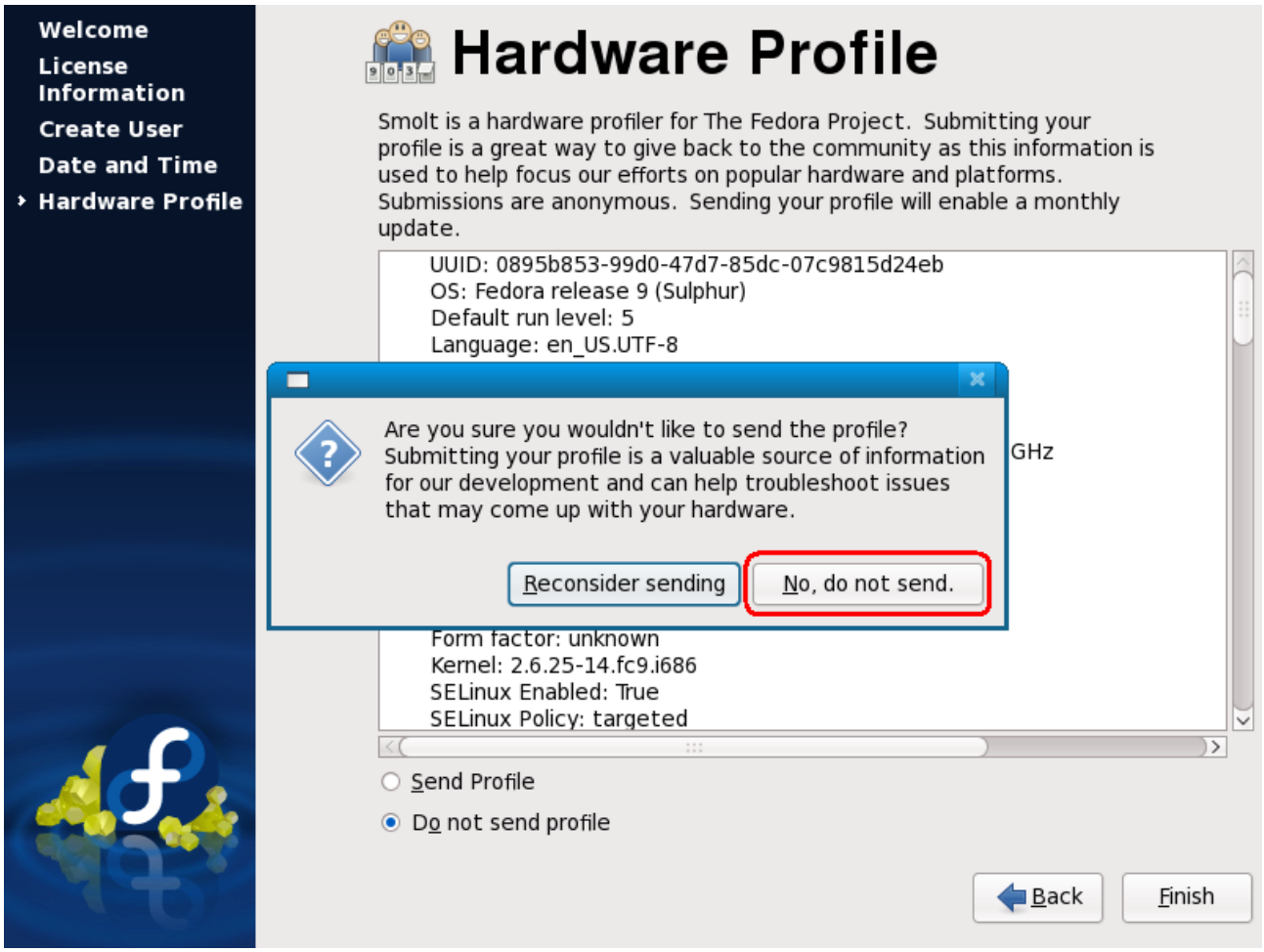
Step18: setup date and time. We ignored this and went to the next step.



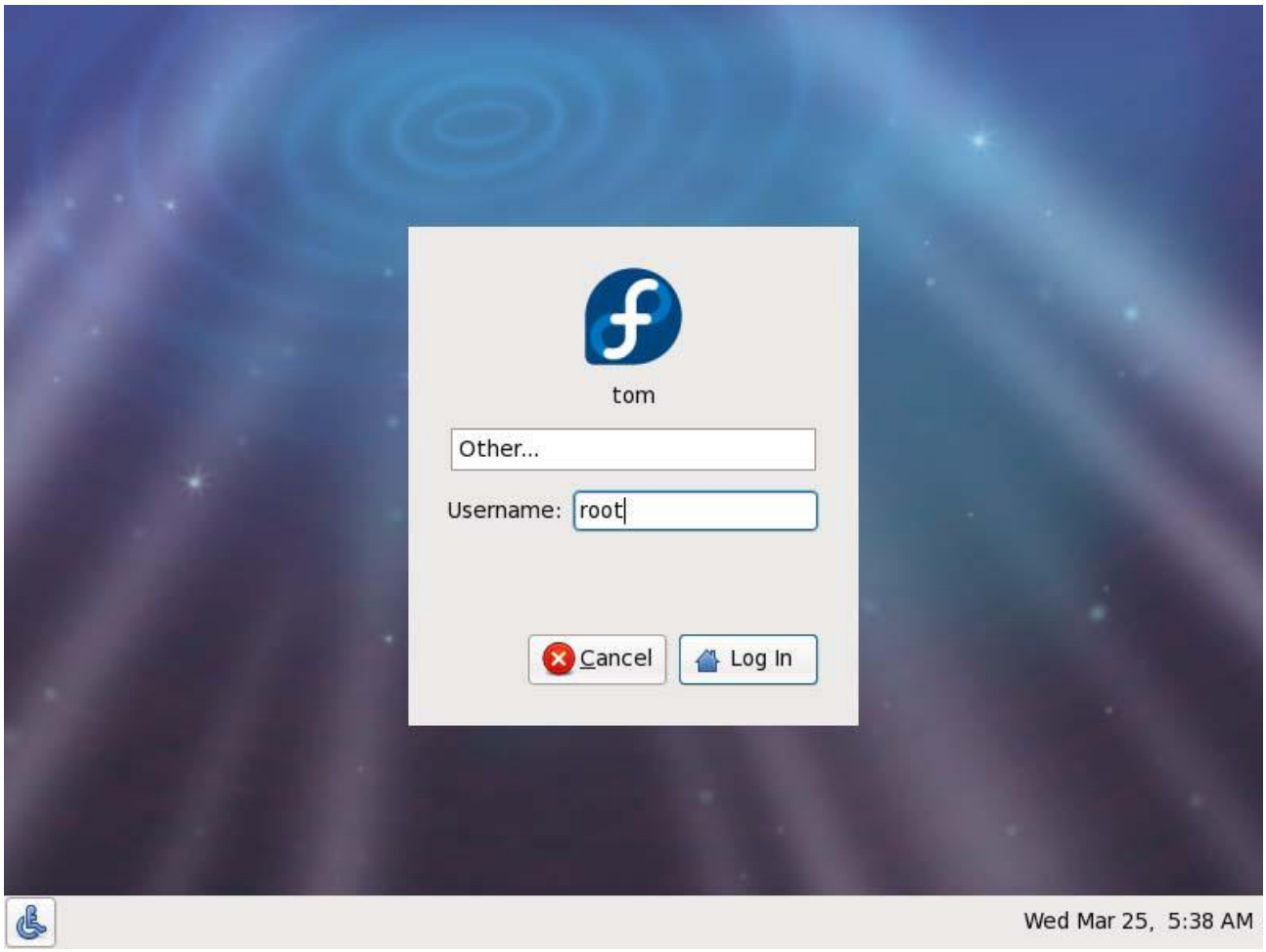
Step19: confirm hardware information. We just clicked on “Finish”.



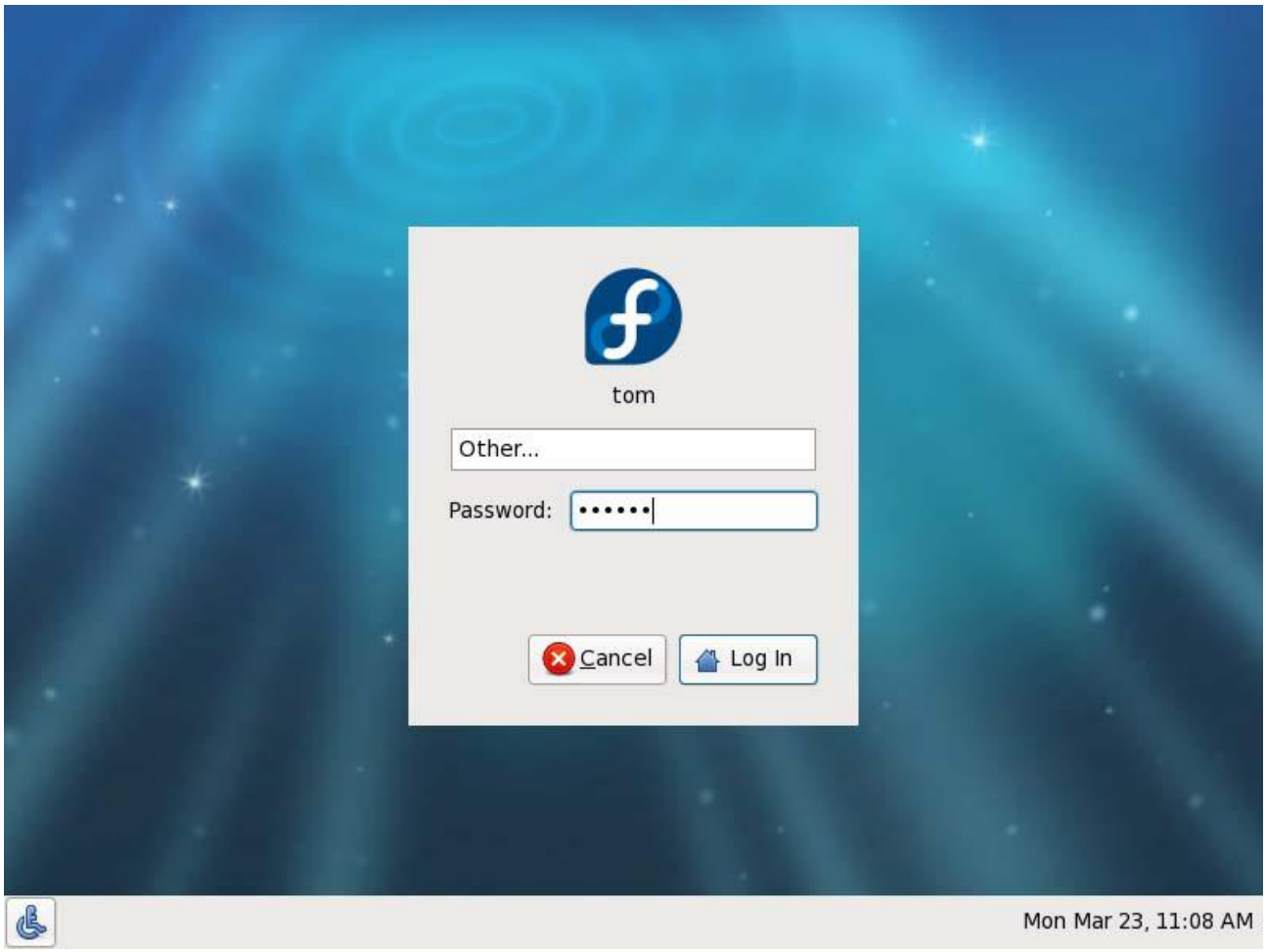
On the popup window shown below, just click on the red marked button.



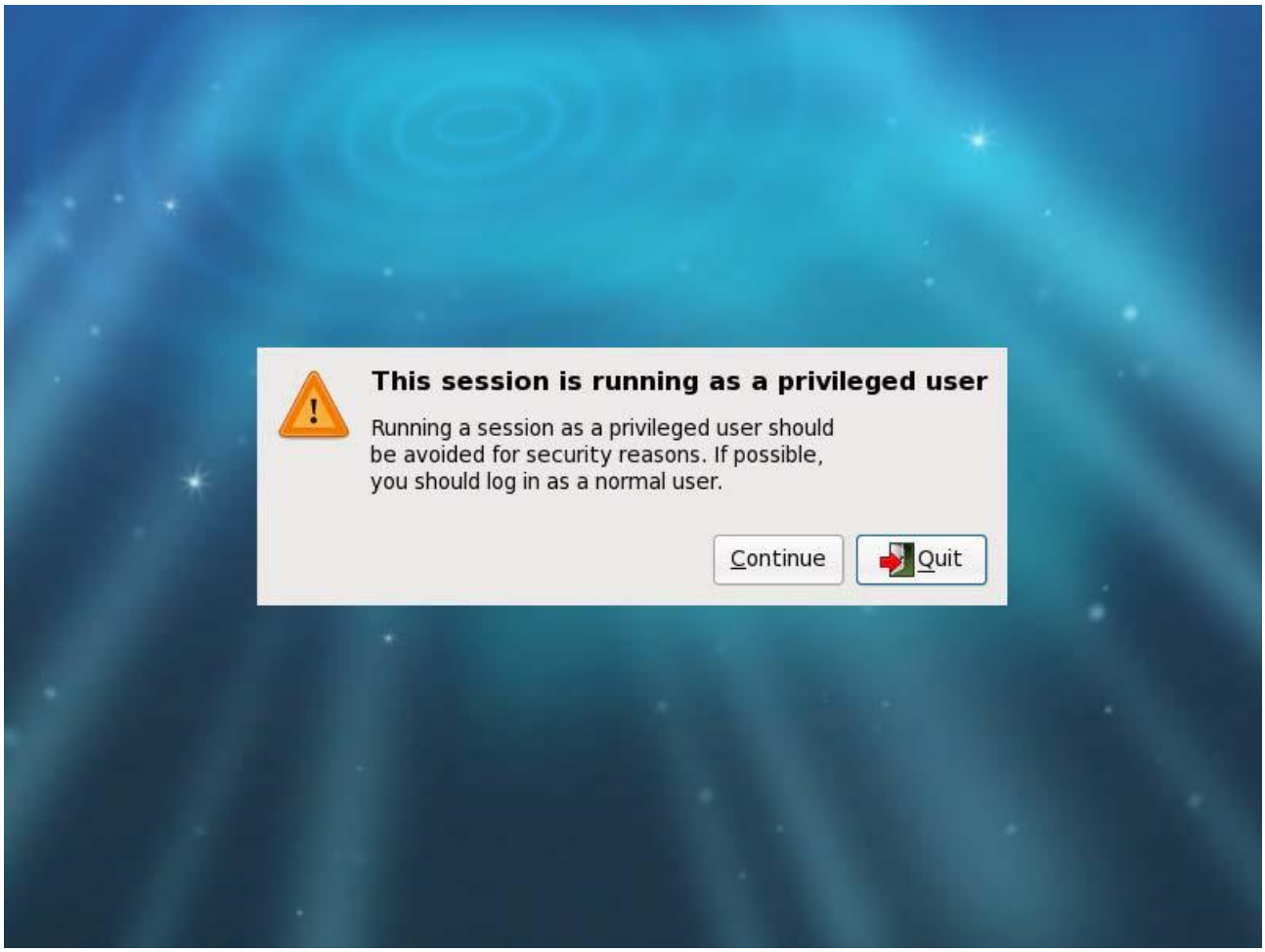
Step 20: on the login page, login as “root”



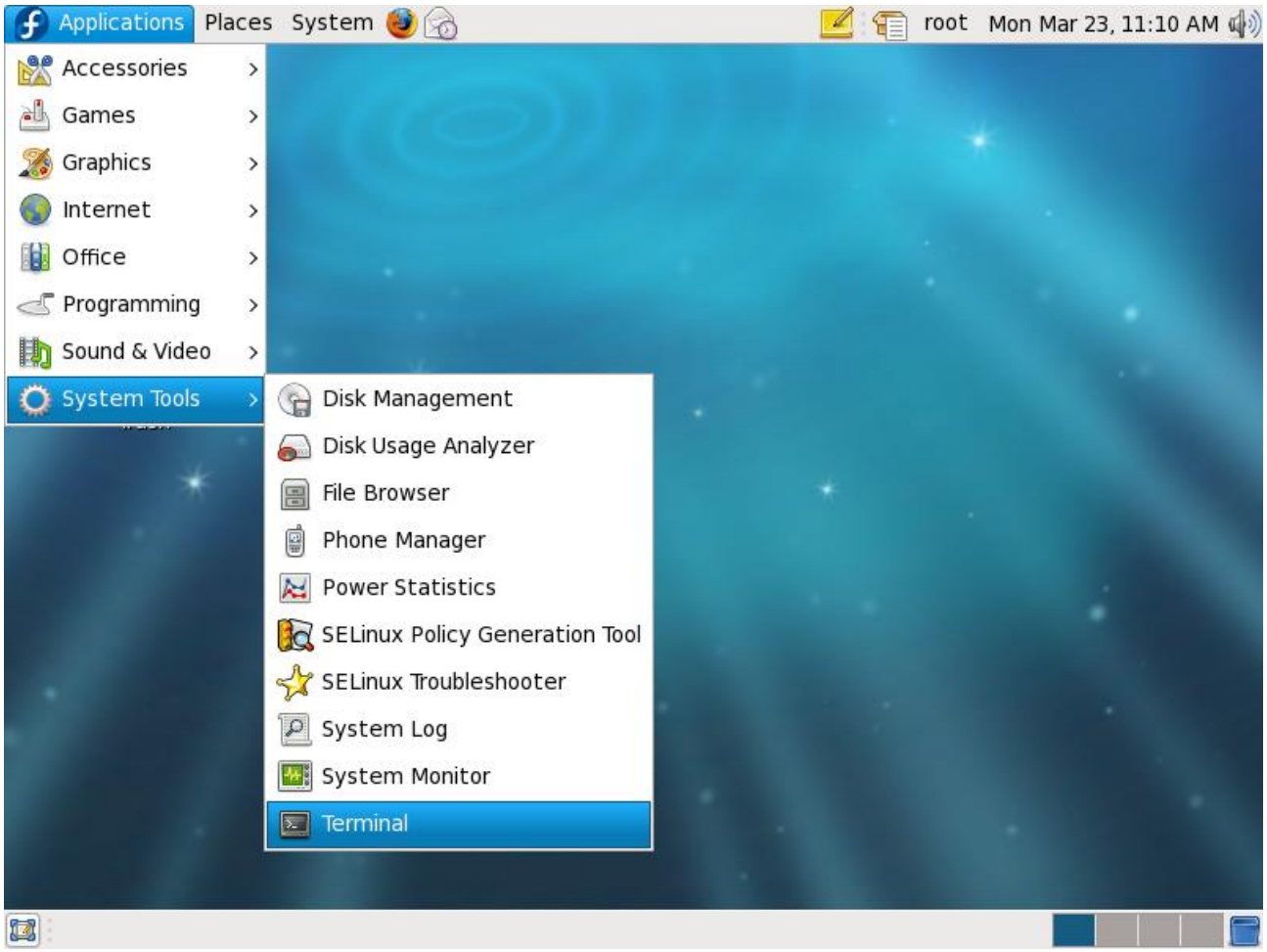
Input the password we just created for “root”



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



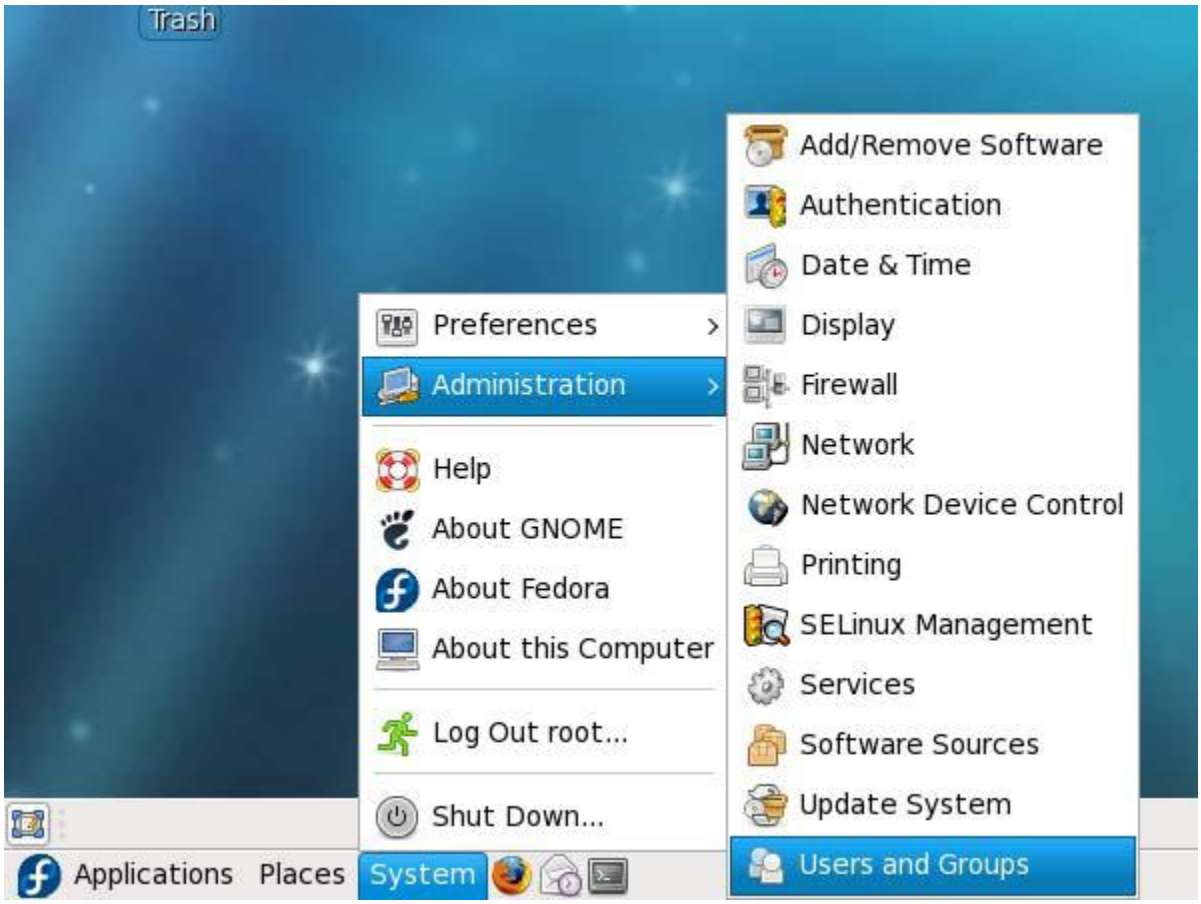
Below is the interface the user will see after a successful login.



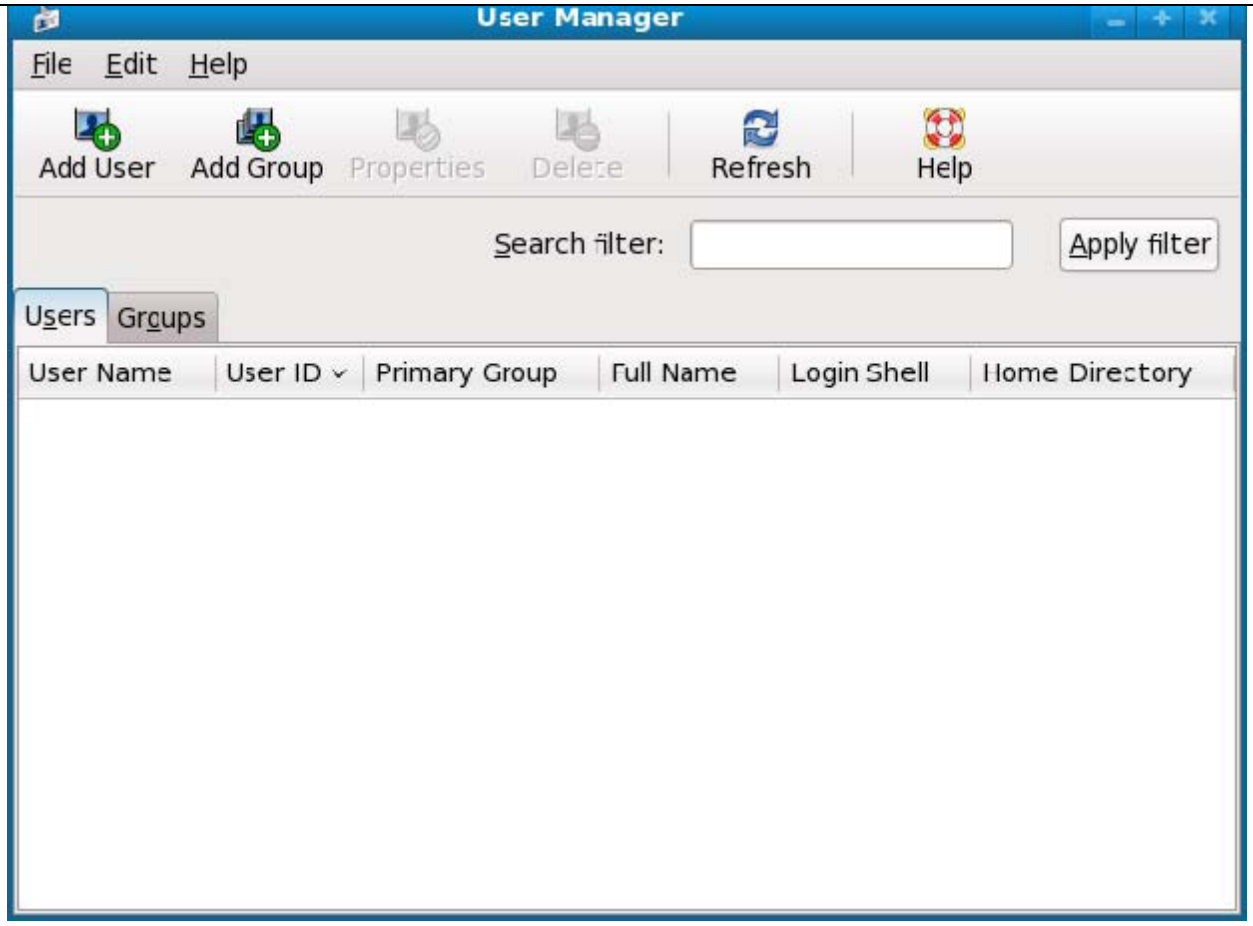
3.2 Add User Account

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

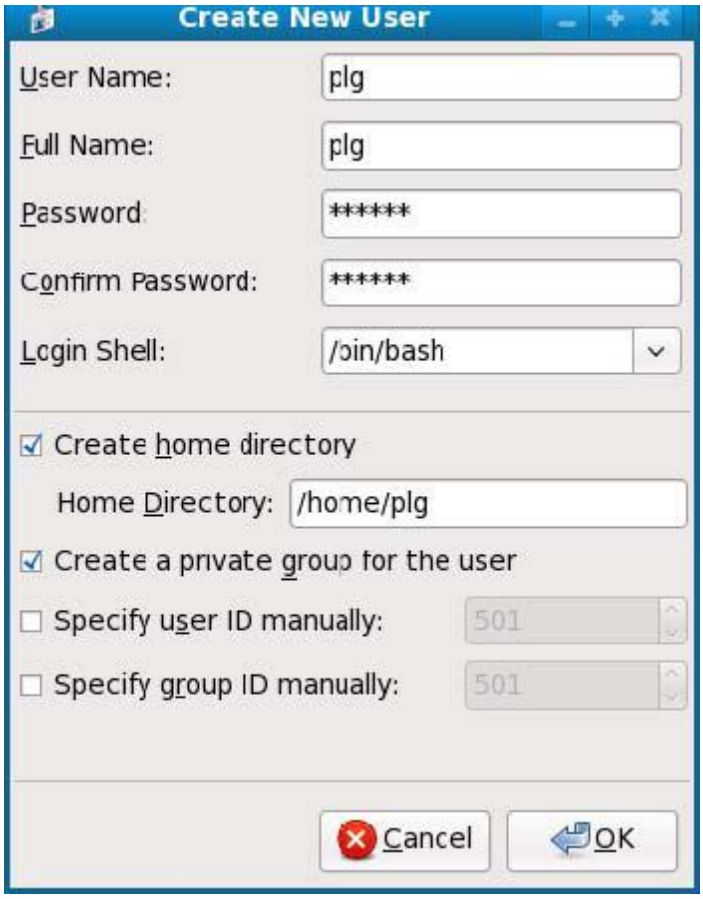
Step 1: go to “Users and Groups”



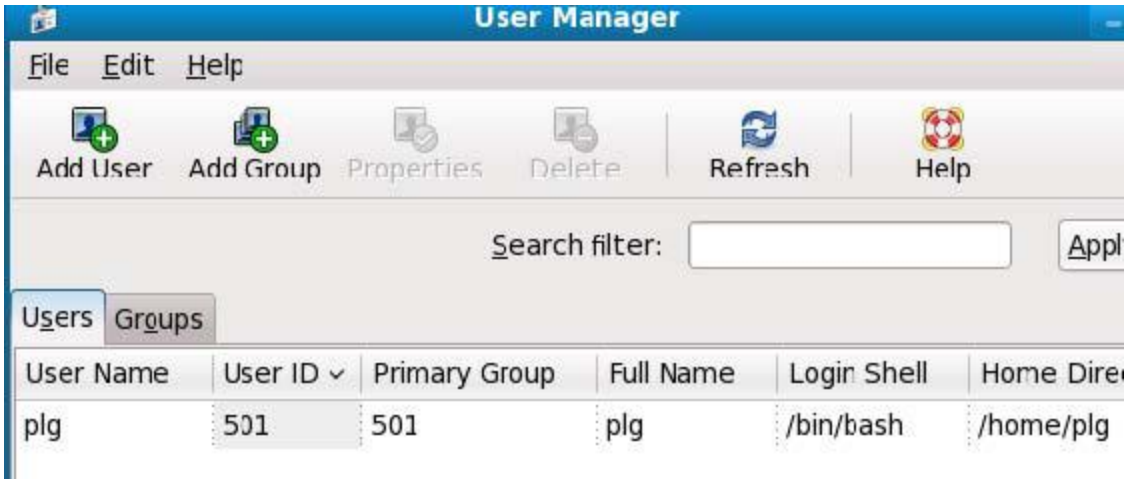
Step 2: open the “Users Manager” window

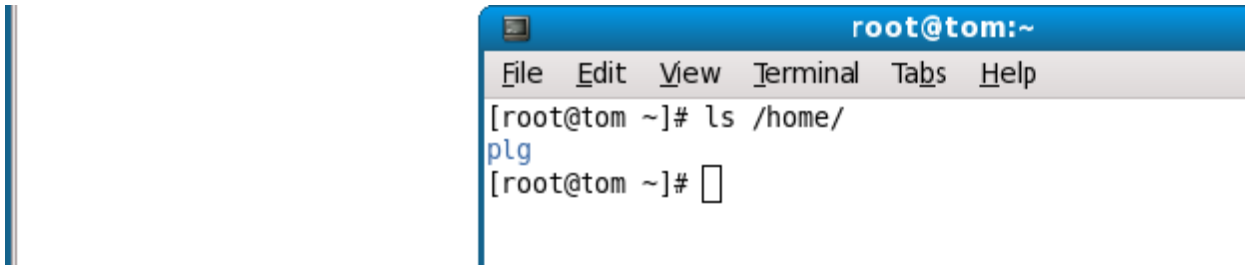


Step 3: click on the “Add User” button, type the user name and password



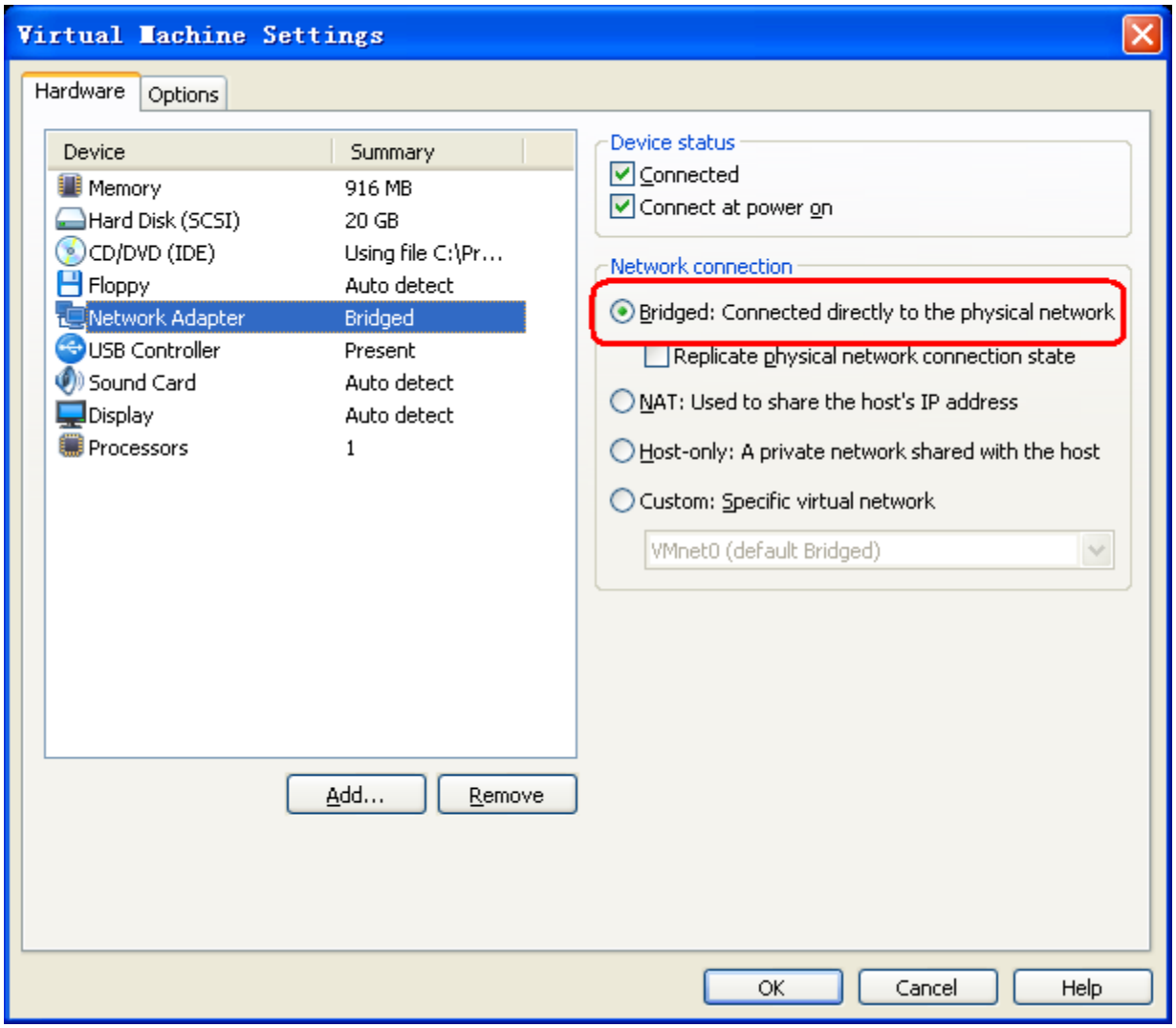
Click on “OK”, you will see that a new “plg” user has been created, and a “plg” directory has been created in the “/home” directory too.





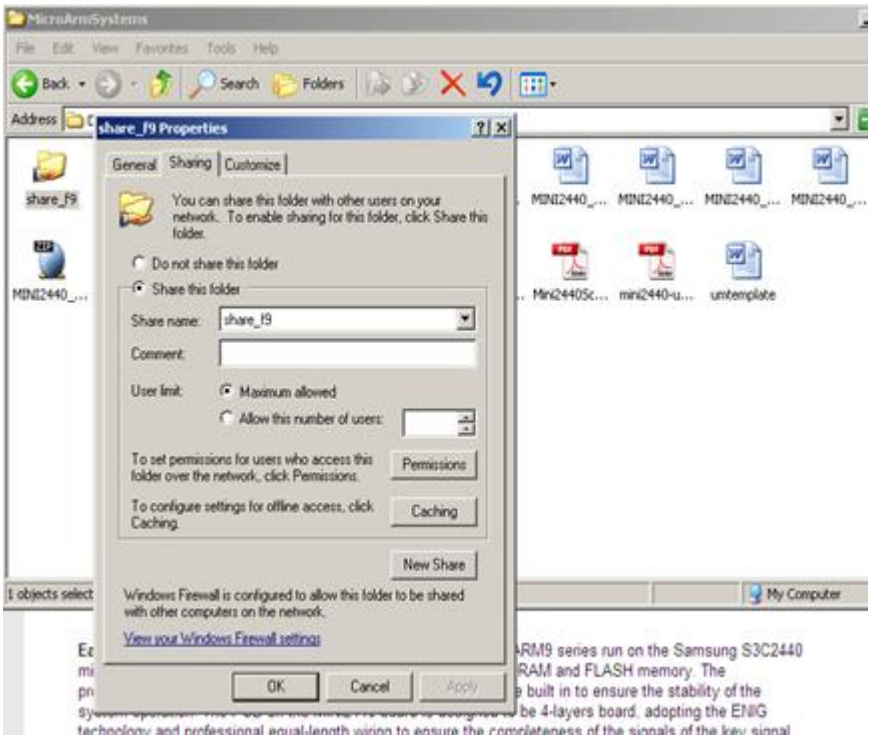
3.3 Access Windows Files

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.

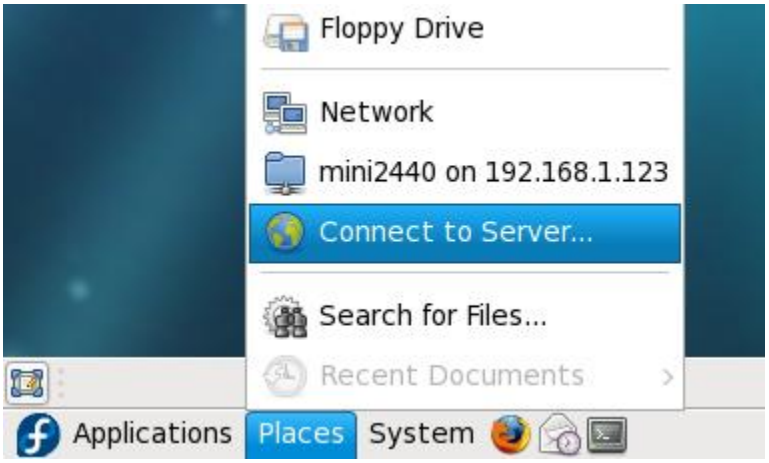


To access shared files in Windows, please following the steps below:

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



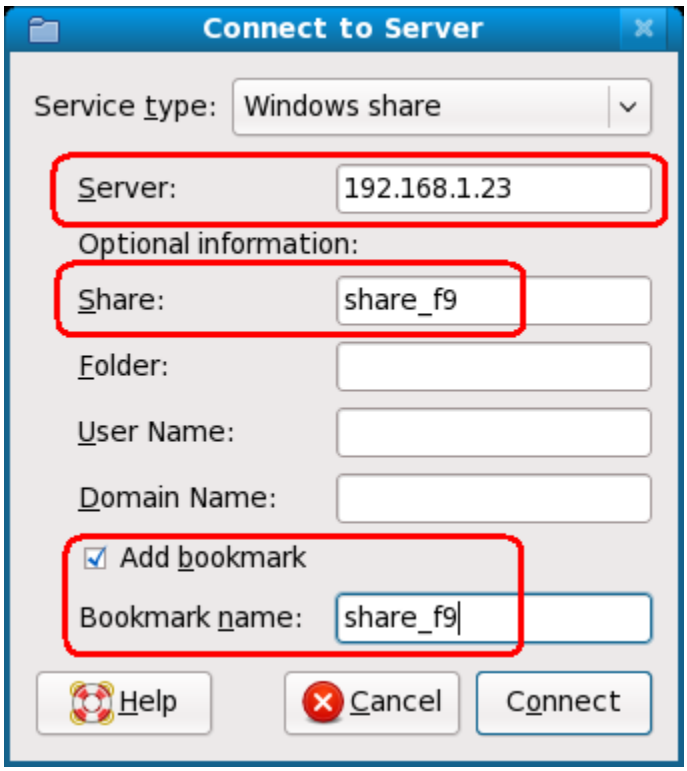
Step 2: set Fedora9



Open the window below:

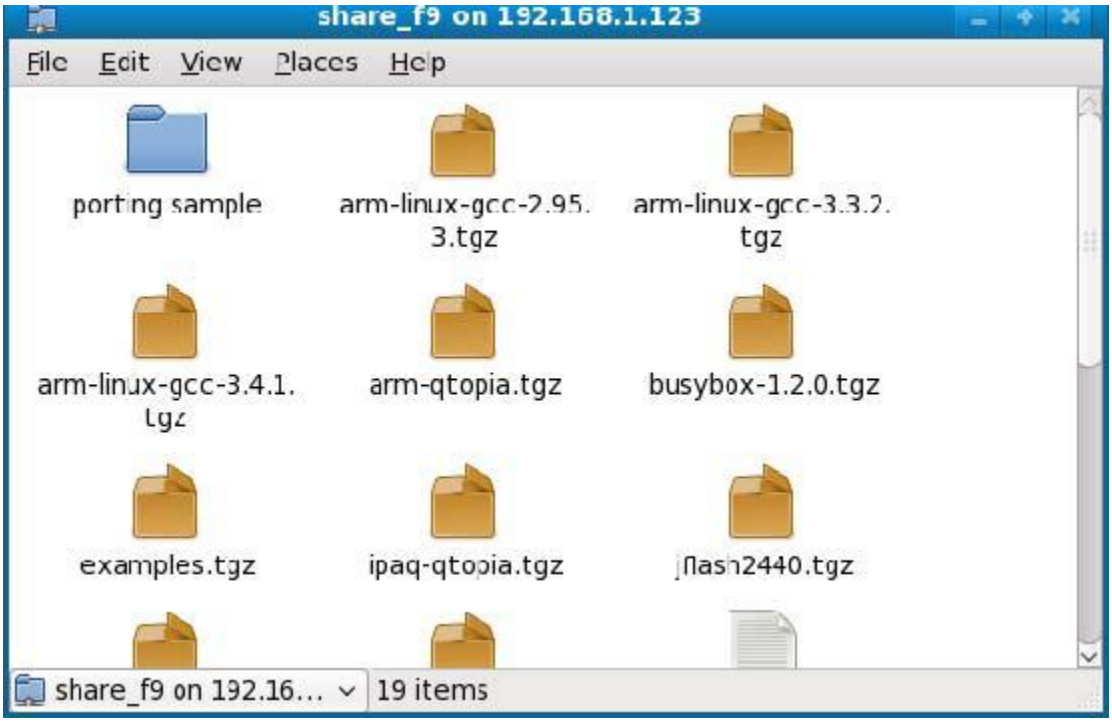
Select “Windows share” in the “service type” field

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

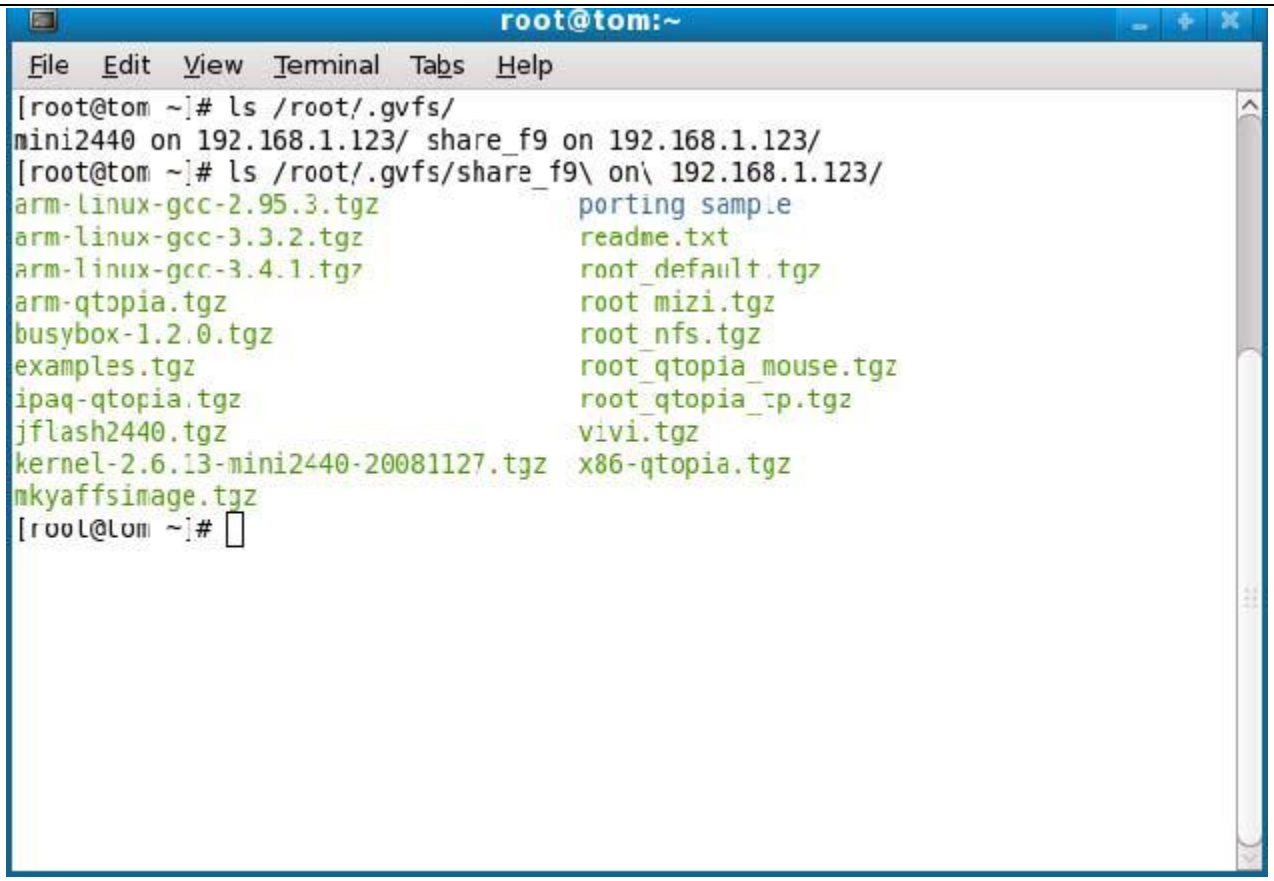


Click on “connect”, the following window will show up:

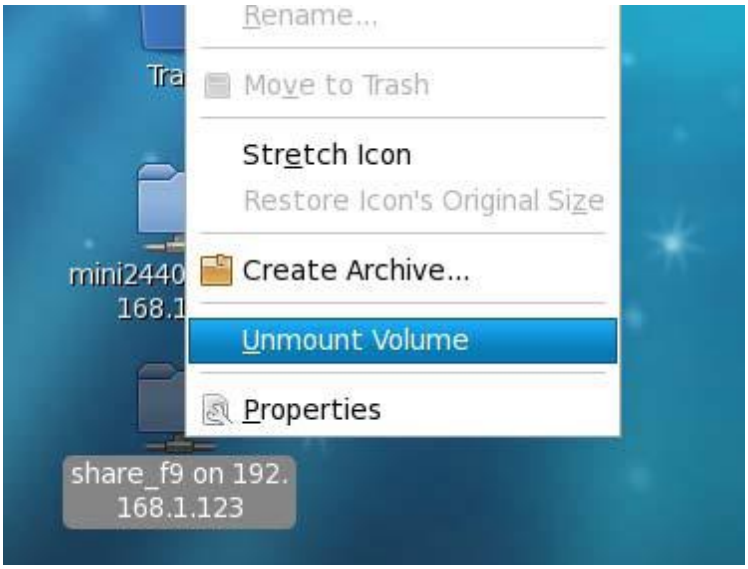
Go ahead and “connect” again, you will see the shared files you just set in your windows system.



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:



3.4 Configure NFS Service

If you have installed Fedora9 on your system, all the corresponding NFS components will be installed by default, you can just follow the steps below to setup and configure the NFS service.

Step 1: Setting Up a Shared Directory

Note: to access a shared directory, you need to follow what were described in 4.2 to install the target file system.

(1) Set up Shared Directories

Run the command below:

#gedit /etc/exports

This command edits the NFS configuration file. Add the following line (*Note：if this file is opened for the first time, it will be empty*):

```
/opt/FriendlyARM/Mini6410/linux/root_qtopia_qt4
*(rw,sync,no_root_squash)
```

"**/opt/FriendlyARM/mini6410/linux/root_qtopia_qt4**" is a NFS shared directory, it can be mounted as the root file system through NFS;

* means all clients can mount to this directory.

"rw" means all clients that have been mounted to this directory have the read and write rights to this directory.

"no_root_squash" means all clients that have been mounted to this directory can be set to a root user

Step 2: Starting NFS

You can start the NFS service through either command line or graphic interface. We set up the NFS service to let others access shared directories. By default Fedora starts it firewall which will disable the NFS service. So you need to disable the firewall by typing "lokkit" in a command line window.

Select (*) Disabled, and click on the "OK" button to disable the firewall permanently. Now you can start the NFS service:

(1) Start and Stop the NFS service

Run the command below:

#/etc/init.d/nfs start

This command will start the NFS service. The user can verify whether the service is running by commanding:

mount -t nfs localhost: **/opt/FriendlyARM/mini6410/linux/root_qtopia_qt4** /mnt/

If no err messages come up, the user can then browse the contents of the "**/mnt**" directory and verify if the contents are the same as the "**/opt/FriendlyARM/mini6410/linux/root_qtopia_qt4**" directory.

Stop the service by commanding:

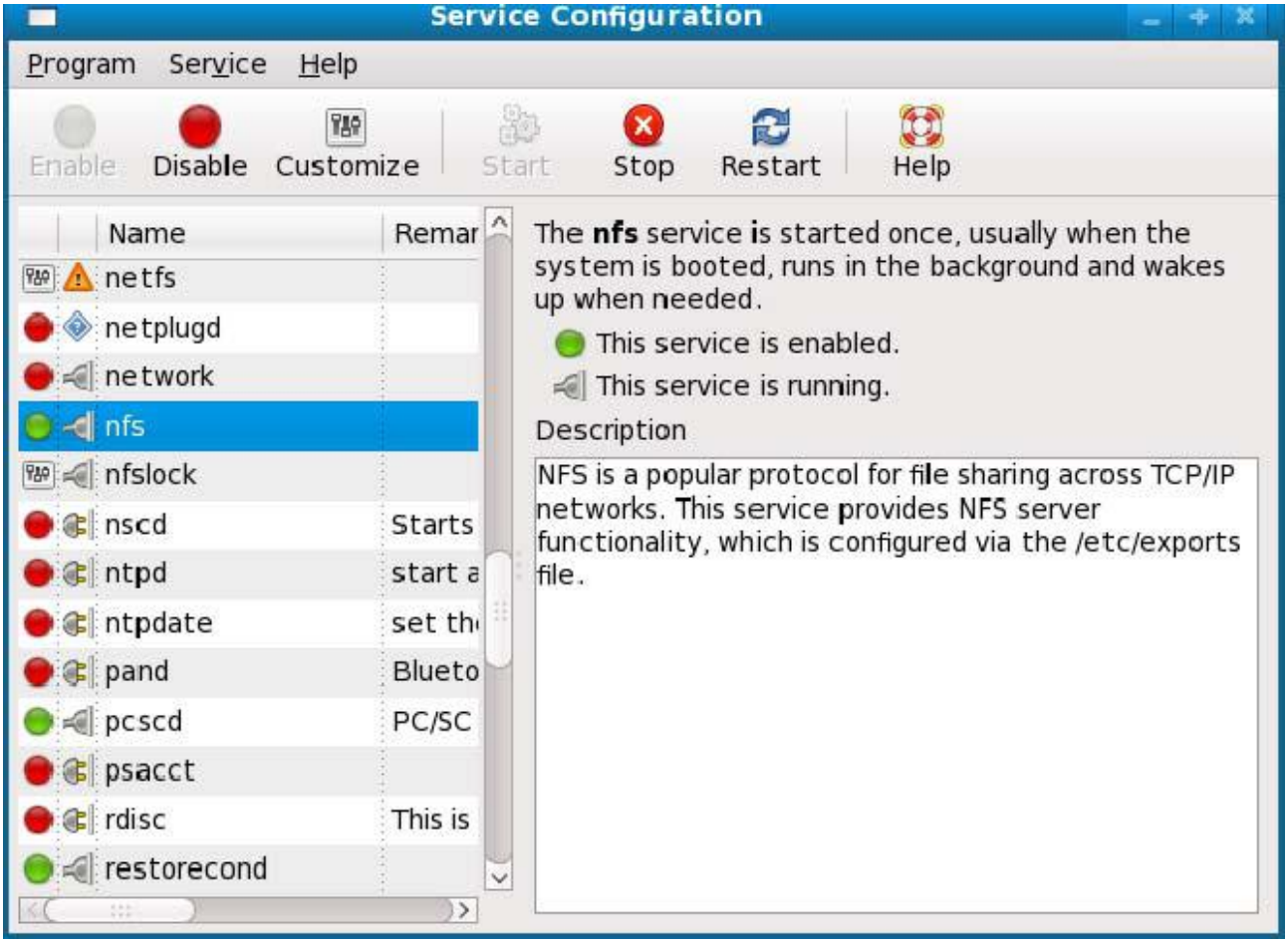
#**/etc/init.d/nfs stop**

(2) Starting the NFS service through the graphic interface

To auto run the service on system startup, the user can execute the command below:

 # serviceconf

Open the system configuration window, on the left side of the window, check the NFS box, click on the "Enable" button to start it.



Step 3: Booting System via NFS

After setting up and running the NFS service, the user can set the NFS as the root file system to boot the board. To boot the system via NFS, the board can fully utilize a "big" hard disk because the user can use the host PC's hard disk, this trick is widely used in Linux development.

Switch the target board's boot mode to the "SDBOOT" side (note: you need to enter the super terminal menu, please refer to 2.4), connect the power cable, serial port cable and the network cable, and open a super terminal. Type the following command:

console=ttySAC0 root=/dev/nfs

nfsroot=192.168.1.111:/opt/FriendlyARM/mini6410/linux/root_qtopia_qt4

ip=192.168.1.70:192.168.1.111:192.168.1.111:255.255.255.0:mini6410.arm9.net:eth0:off

"nfsroot" is the board's IP. If you started a virtual machine this IP would be your virtual machine's IP.

The number strings after "ip=" are detailed as below:

The first item, in this example "192.168.1.70" is the target's temporary IP (please make sure this IP doesn't conflict with other IPs within the same network);

The second item, in this example "192.168.1.111" is the host's IP,

The third item, in this example "192.168.1.111" is the target board's gateway IP,

The fourth item, in this example "255.255.255.0" is the subnet mask,

The fifth item is the board's machine name (the user can give whatever name he likes)

"eth0" is the network adaptor's name.

This command is so long that it could be easily typed wrong. In this shipped CD, this command has been written in the "nfs.txt" file for the customer's convenience. The user can copy it directly. After "enter" these parameters will be saved in the NAND Flash.

```
[l] Download WinCE bootlogo
[w] Download WinCE NK.bin
[b] Boot the system
[s] Set the boot parameter of Linux
[i] Version: 1022
Enter your Selection:s
Linux cmd line: console=ttySAC0 root=/dev/nfs nfsroot=192.168.1.111:/opt/FriendlyARM/mini6410/root_qtopia_qt4 ip=192.168.1.70:192.168.1.111:192.168.1.111:255.255.255.0:sbc2440.arm9.net:eth0:off"
Linux command line saved
###### FriendlyARM Superboot(6410) for 6410 ######
[f] Format the nand flash
[v] Download uboot.bin
[k] Download Linux/Android kernel
[y] Download root yaffs2 image
[u] Download root ubifs image
[a] Download Absolute User Application
[n] Download Nboot.nb0 for WinCE
[l] Download WinCE bootlogo
[w] Download WinCE NK.bin
[b] Boot the system
[s] Set the boot parameter of Linux
[i] Version: 1022
Enter your Selection:_
```

Then type “b” and press “enter” to boot the system via NFS.

```
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
s3c2410-rtc s3c2410-rtc: setting system clock to 2000-01-01 23:43:15 UTC (946770195)
eth0: link down
IP-Config: Complete:
     device=eth0, addr=192.168.1.70, mask=255.255.255.0, gw=192.168.1.111,
     host=sbc2440, domain=, nis-domain=arm9.net,
     bootserver=192.168.1.111, rootserver=192.168.1.111, rootpath=
Looking up port of RPC 100003/2 on 192.168.1.111
eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
Looking up port of RPC 100005/1 on 192.168.1.111
VFS: Mounted root (nfs filesystem).
Freeing init memory: 124K
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case sensitive!
[01/Jan/2000:15:43:29 +0000] boa: server version Boa/0.94.13
[01/Jan/2000:15:43:29 +0000] boa: server built Apr  8 2010 at 15:40:06.
[01/Jan/2000:15:43:29 +0000] boa: starting server pid=654, port 80

Try to bring eth0 interface up......NFS root ...Done

Please press Enter to activate this console.
[root@FriendlyARM /]# _
```

```
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
s3c2410-rtc s3c2410-rtc: setting system clock to 2000-01-01 23:43:15 UTC (946770
195)
eth0: link down
IP-Config: Complete:
     device=eth0, addr=192.168.1.70, mask=255.255.255.0, gw=192.168.1.111,
     host=sbc2440, domain=, nis-domain=arm9.net,
     bootserver=192.168.1.111, rootserver=192.168.1.111, rootpath=
Looking up port of RPC 100003/2 on 192.168.1.111
eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
Looking up port of RPC 100005/1 on 192.168.1.111
VFS: Mounted root (nfs filesystem).
Freeing init memory: 124K
FAT: utf8 is not a recommended IO charset for FAT filesystems, filesystem will b
e case sensitive!
[01/Jan/2000:15:43:29 +0000] boa: server version Boa/0.94.13
[01/Jan/2000:15:43:29 +0000] boa: server built Apr  8 2010 at 15:40:06.
[01/Jan/2000:15:43:29 +0000] boa: starting server pid=654, port 80

Try to bring eth0 interface up......NFS root ...Done

Please press Enter to activate this console.
[root@FriendlyARM /]# _
```

3.5 Set Up Cross Compile Environment

To compile kernels, Qtopia/Qt4, bootloader and other programs in Linux you need a a cross compile environment. We used arm-linux-gcc-4.5.1 and its by defauly supports armv6 command sets. The following steps will introduce how to build a compile environment.

Step 1: copy the compressed file “arm-linux-gcc-4.5.1-v6-vfp-20101103.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-4.5.1-v6-vfp-20101103.tgz –C /

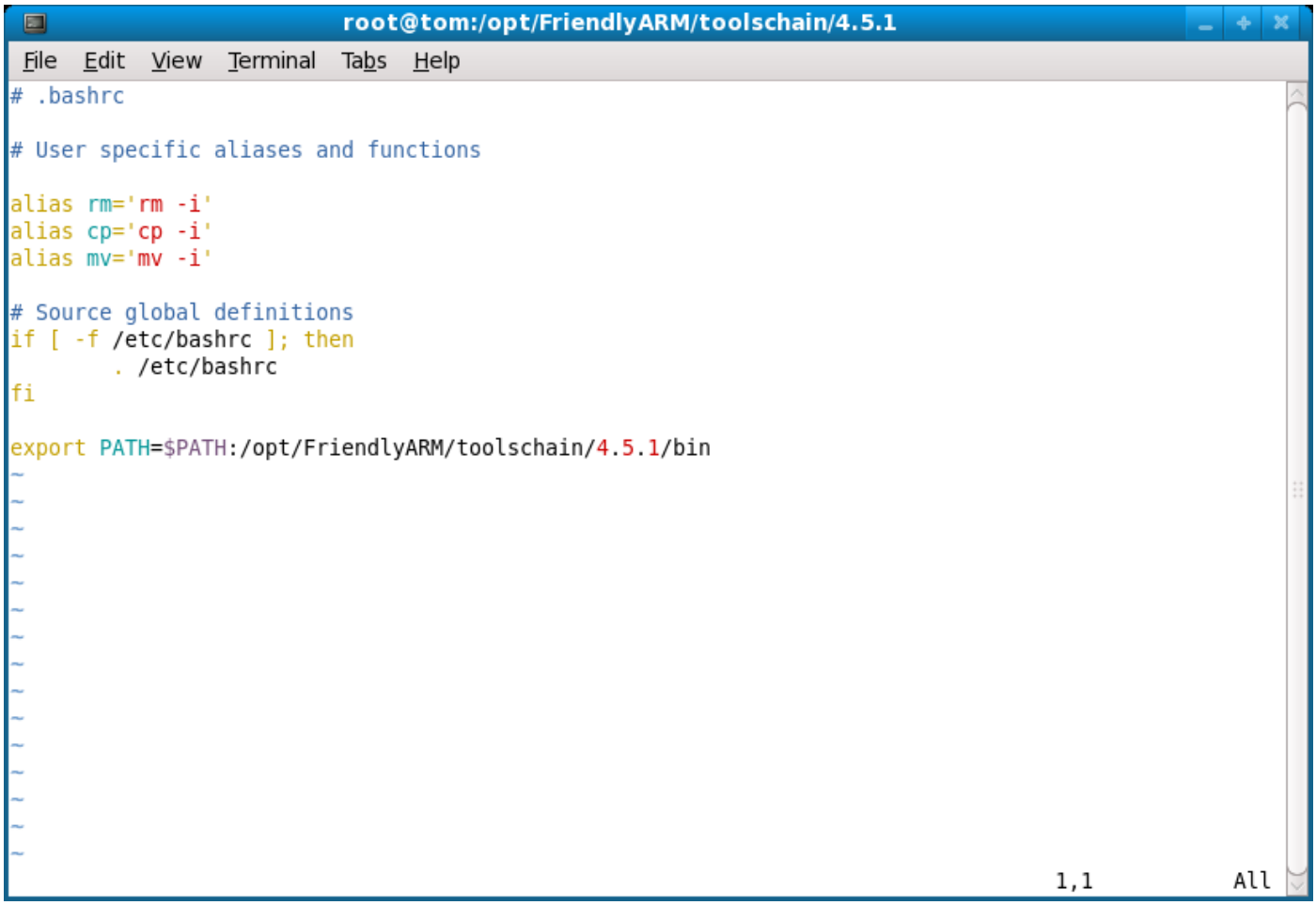
Note: there is a space after "C" and "C" is a capital letter.

These commands will install "arm-linux-gcc" in the "**/opt/FriendlyARM/toolschain/4.5.1**"

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

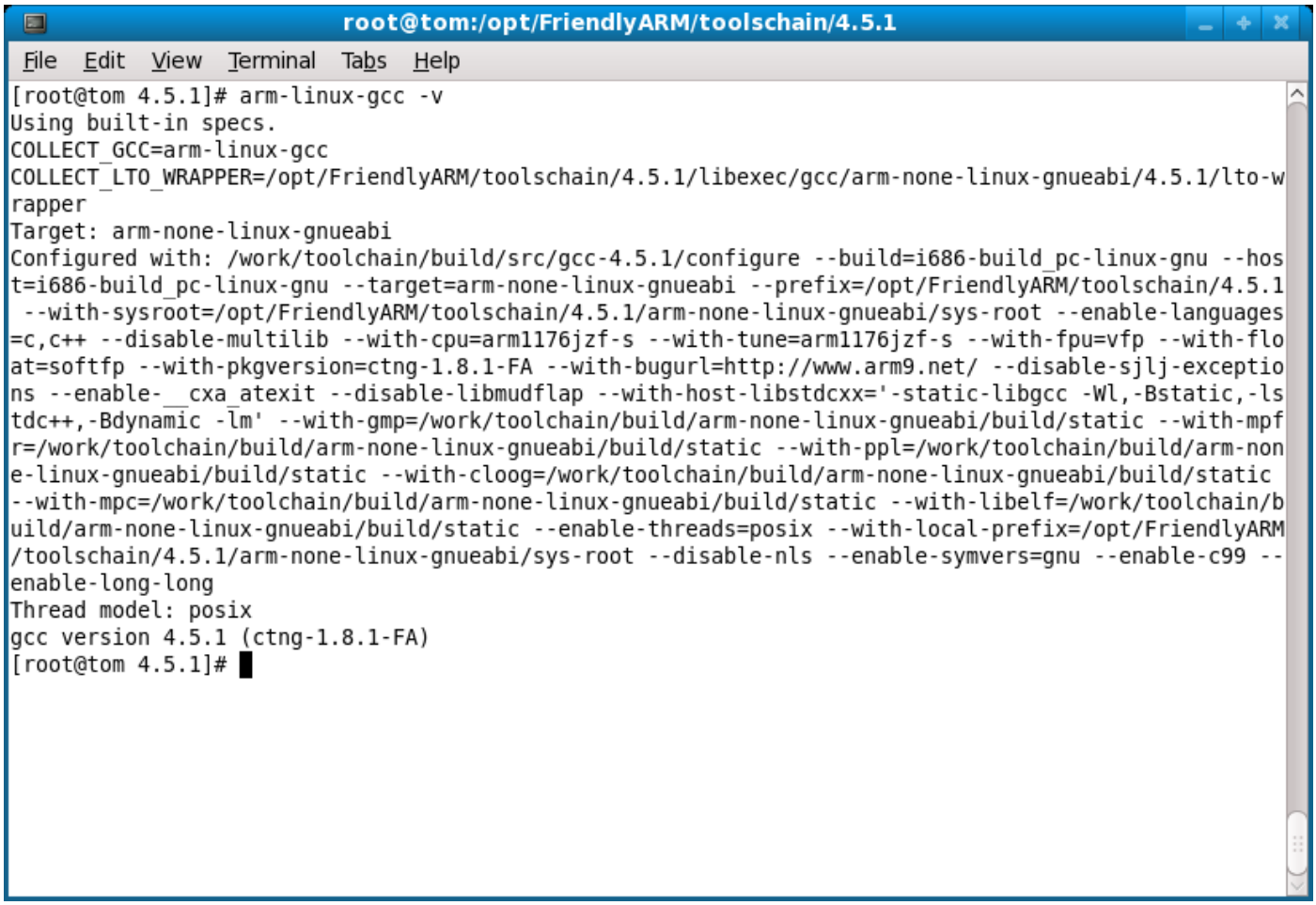
#gedit /root/.bashrc

This is to edit the "/root/.bashrc" file. Update the last line with "**export PATH=$PATH:/opt/FriendlyARM/toolschain/4.5.1/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.



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)
- arm-qt-extended-4.4.3 source code (i.e. Qtopia4, for x86 and arm）
- QtE-4.7.0 (for ARM)
- Busybox-1.17 source code
- Sample programs code (developed by FriendlyArm)
- U-Boot
- 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-4.4.1

4.1 Uncompress Source Code

Firstly, create a working directory: /opt/FriendlyARM/mini6410/linux

After execute command "mkdir –p /opt/FriendlyARM/mini6410/linux", 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/linux" by running the following command

#mkdir /tmp/linux

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

(2) Uncompress and install the u-boot source code

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/u-boot-mini6410-20101106.tar.gz

A u-boot-mini6410 directory will be created and it includes a complete copy of u-boot source code.

Note: 20101106 is the date when FriendlyARM released the new version, the file name in the shipped CD may be different.

(3) Uncompress the Linux kernel source code

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/linux-2.6.36-20101115.tar.gz

A linux-2.6.36 directory will be created, it includes a complete copy of linux kernel source code.

Note: 20101115 is the date when FriendlyARM released the new version, the file name in the shipped CD may be different.

(4) Uncompress and Install the target file system

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/rootfs_qtopia_qt4-20101120.tgz

A rootfs_qtopia-qt4 directory will be created, it includes a complete copy of linux kernel source code.

Note: 20101120 is the date when FriendlyARM released the new version, the file name in the shipped CD may be different.

(5) Uncompress and install Qtopia source code

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/x86-qtopia-29199429.tar.gz

#tar xvzf /tmp/linux/arm-qtopia-20101105.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. And the source code for the embedded browser konquor is included too.

(6) Uncompress and install qt-extended-4.4.3 source code

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/x86-qt-extended-4.4.3-20101003.tgz

#tar xvzf /tmp/linux/arm-qt-extended-4.4.3-20101105.tgz

An x86-qt-extended-4.4.3 and an arm-qt-extended-4.4.3 will be created, and their source code will be uncompressed into these two directories.

(7) Uncompress and install QtE-4.7.0 source code

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/x86-qte-4.6.1-20100516.tar.gz

#tar xvzf /tmp/linux/arm-qte-4.7.0-20101105.tar.gz

An x86-qte-4.6.1 and an arm-qte-4.7.0 will be created, and their source code will be uncompressed into these two directories.

(8) Uncompress and install busybox source code

The Busybox is a compact Linux tool kit. Here we used busybox-1.17.2. Users can download its latest version from http://www.busybox.net

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/busybox-1.17.2-20101120.tgz

A busybox-1.17.2 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.

(9) Uncompress and install Linux sample programs

In the work directory /opt/FriendlyARM/mini6410/linux, run the commands below:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/examples-mini6410-20101110.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.

4.2 Create Target File System

We offered the following two packages:

- rootfs_qtopia_qt4-20101120.tgz
- rootfs_qtopia_qt4-s-20101120.tgz

The one with "-s" is for LCDs that have dedicated touch screen controller such as large size four-wire resistor touch screens and the other is for LCDs that utilize ARM's touch

screen controller. The only difference between the two relies on the configurations in the "/etc/friendlyarm-ts-input.conf" file.

Execute the following commands:

#cd /opt/FriendlyARM/mini6410/linux

#tar xvzf /tmp/linux/ rootfs_qtopia_qt4-20101120.tgz

tar xvzf /tmp/linux/ rootfs_qtopia_qt4-s-20101120.tgz

A rootfs_qtopia_qt4 and a rootfs_qtopia_qt4-s will be created.

This package includes qtopia-2.2.0, Qtopia4 and QtE-4.7.0, 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
- support co-existence of a USB mouse and a touch screen (since Linux-2.6.36)

4.3 Install Target File System

To burn a target file system to the board you need to make the file system an image first.

We offered two tools that can be used to make file images:mkyaffs2image and mkyaffs2image-128M. The “mkyaffs2image” utility is for 64M and the “mkyaffs2image-128M” is for 128M/256M/512M/1GB.

Note: mkyaffs2image and mkyaffs2image-128M are only for SLC Nand Flash to make yaffs2 iamges and don’t apply to MLC2 Nand Flash. The generated image file’s structure is “1 Page= 2K Byte， 1 Block=128K” for SLC Nand Flash(such as Samsung’s K9F2G08，k9K8G08).

For the Mini6410 system we don’t have a 64M system so “**mkyaffs2image-128M**” can be applied here too.

In addition for users’ sake we also developed a **mkubimage-slc** which is especially for making UBIFS for SLC Nand flash. And the “**mkext3image**” is for EXT3. We call all these tools “mktools” uniformly. Below are the steps to install:

#tar xvzf /tmp/linux/mktools.tar.gz –C /

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

4.4 Install LogoMaker

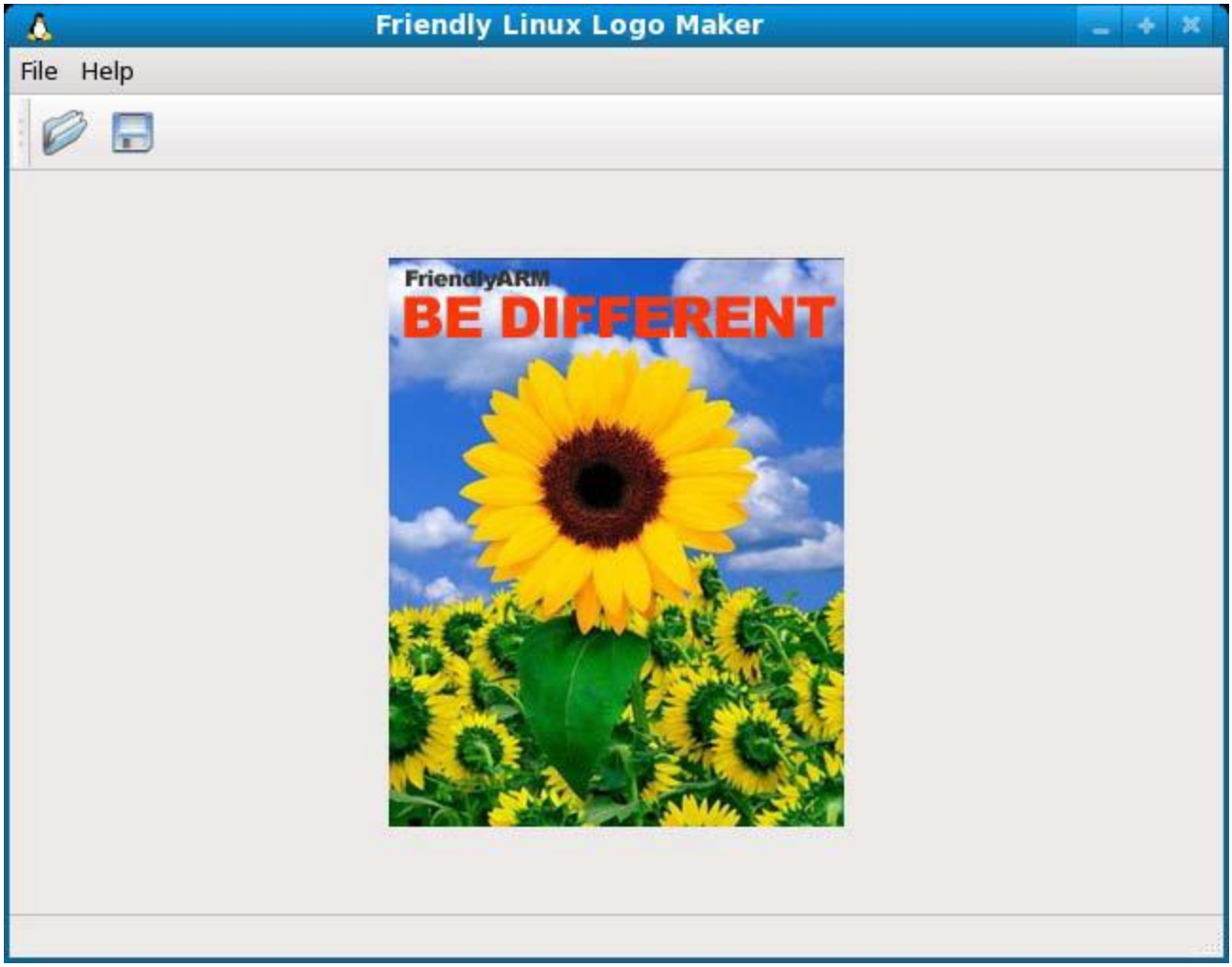
LogoMaker is developed by FriendlyARM for making linux logos. There are many resources describing how to convert image files such as bmp, jpg, png and so on to linux logos using command line tools. We created this graphic version which is based on Fedora9.

Execute the command below:

#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


Friendly Linux Logo Maker
File Help
FriendlyARM
BE DIFFERENT

5 Configure and Compile U-Boot

Samsung has migrated a U-Boot for 6410 and it supports USB download, NAND booting. It is open source. We have made some modifications based on it:

- Add a download menu similar to Superboot's USB download menu
- Support SD booting configuration
- Support direct burning of YAFFS2
- Support burning of WindowsCE's BootLoader nboot
- Support burning of WindowsCE image
- Support burning of standalone programs
- Support returning to the start shell
- Support 256M DDR RAM
- Support boot MLC Nand Flash(but this u-boot utility can't write MLC NAND Flash)

Below are the steps on how to configure and compile it:

5.1 Configure and Compile U-Boot that Supports NAND Booting

Note: different DDR RAM systems require differed u-boot configurations.

To compile a u-boot for 128M RAM please follow the steps below:

Enter the U-boot source code directory and execute:

#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410

#make mini6410_nand_config-ram128;make

This will configure and compile a u-boot.bin that supports the NAND booting. Download it to your board's NAND flash and you will be able to use it. Here we named it "u-boot_nand-ram128.bin" to distinguish from the one in the shipped CD.

To compile a u-boot for 256M RAM please follow the steps below:

Enter the U-boot source code directory and execute:

#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410

#make mini6410_nand_config-ram256;make

This will configure and compile a u-boot.bin that supports the NAND booting. Download it to your board's NAND flash and you will be able to use it. Here we named it "u-boot_nand-ram256.bin" to distinguish from the one in the shipped CD.

5.2 Configure and Compile U-Boot that Supports SD Card Booting

Note: different DDR RAM systems require differed u-boot configurations.

To compile a u-boot for 128M RAM please follow the steps below:

Enter the U-boot source code directory and execute:

#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410

#make mini6410_sd_config-ram128;make

This will configure and compile a u-boot.bin that supports the SD card booting. Download

it via SD-Flasher.exe to your board's SD card, switch your board to "SDBOOT" and you will be able to use it. Here we named it "u-boot_sd-ram128.bin" to distinguish from the one in the shipped CD.

To compile a u-boot for 256M RAM please follow the steps below:

Enter the U-boot source code directory and execute:

#cd /opt/FriendlyARM/mini6410/linux/u-boot-mini6410

#make mini6410_sd_config-ram256;make

This will configure and compile a u-boot.bin that supports the SD card booting. Download it via SD-Flasher.exe to your board's SD card, switch your board to "SDBOOT" and you will be able to use it. Here we named it "u-boot_sd-ram256.bin" to distinguish from the one in the shipped CD.

5.3 Run U-boot

Coming soon!

6 Configure and Compile Kernel

For different LCD systems we offered their corresponding configuration files:

config_mini6410_x35 – for Sony 3.5”LCD, resolution 240x320

config_mini6410_w35 – for TFT landscape 3.5”LCD, resoluiotn 320x240

config_mini6410_n43 – for NEC4.3”LCD, resolution 480x272

config_mini6410_l80 - for Sharp 8” (or compatible models)LCD, resolution 640x480

config_mini6410_a70 – for 7” true color screen, resolution 800x480

config_mini6410_vga1024x768 – for VGA module, resolution 1024x768

config_mini6410_vga800x600 – for VGA module, resolution 800x600

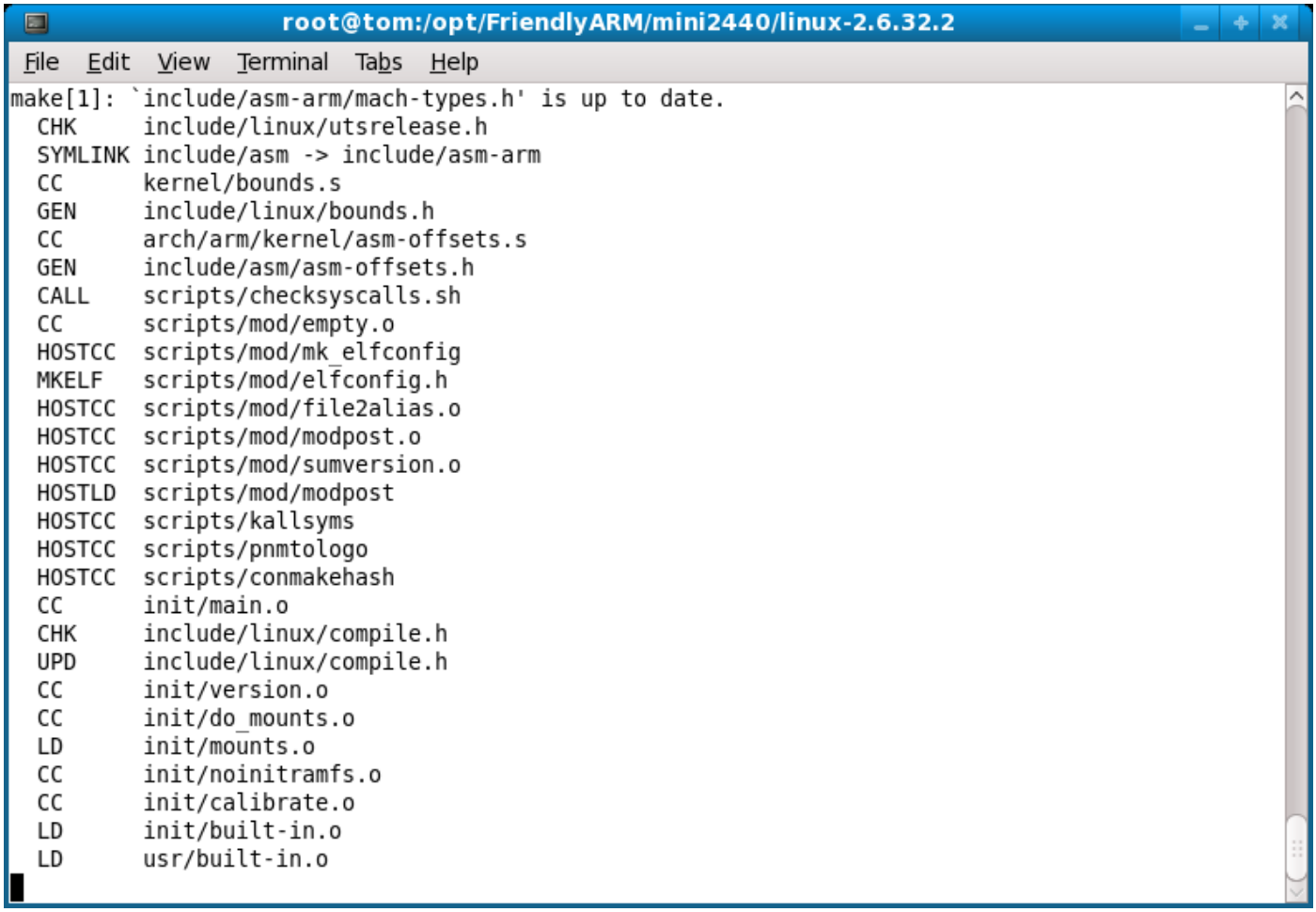
config_mini6410_vga640x480 – for VGA module, resolution 640x480

config_mini6410_ezvga800x600 –for simple VGA module, resolution 800x600

Type the following command to compile:

#cp config_mini6410_n43 .config : there is a space after n43 and a “.” prior to “config”

#make zImage ; begins to compile



After the compilation is done, an image file **zImage** will be generate under "**arch/arm/boot**". There are image files under "images/linux" in the shipped CD such as zImage_n43, zImage_a70 and so on

7 Configure and Compile Busybox

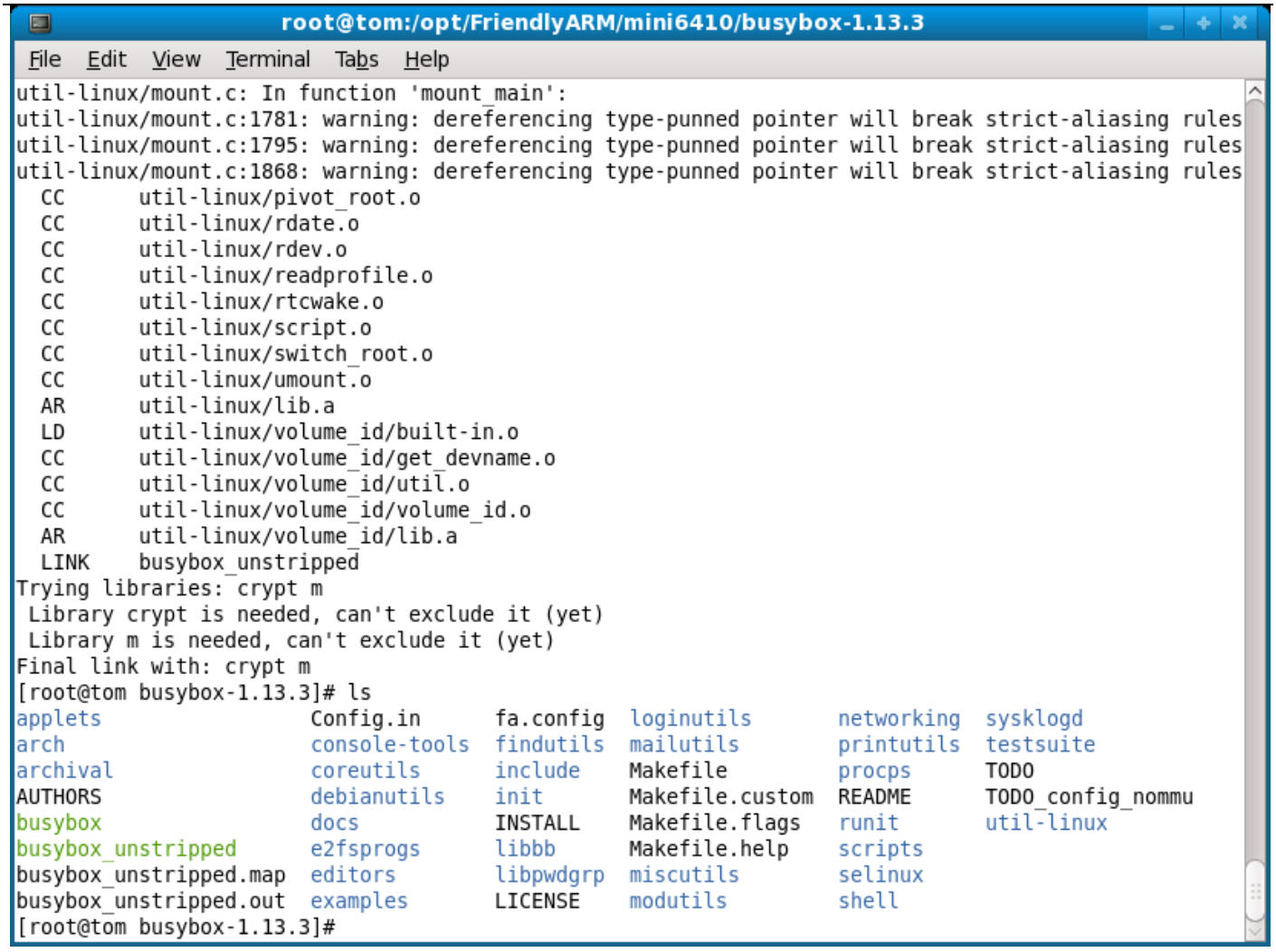
In general a direct download of busybox from its website may be compiled through and its configurations need to be modified. We made one for users: fa.config. Both our 2440 and 6410 systems use this file. Please follow the steps below to install.

Enter the busybox directory:

#cp fa.config .config

#make

A moment later an object file of **busybox** will be generated, which is identical to the one preinstalled in the system


root@tom:/opt/FriendlyARM/mini6410/busybox-1.13.3
File Edit View Terminal Tabs Help
util-linux/mount.c: In function 'mount_main':
util-linux/mount.c:1781: warning: dereferencing type-punned pointer will break strict-aliasing rules
util-linux/mount.c:1795: warning: dereferencing type-punned pointer will break strict-aliasing rules
util-linux/mount.c:1868: warning: dereferencing type-punned pointer will break strict-aliasing rules
CC util-linux/pivot_root.o
CC util-linux/rdate.o
CC util-linux/rdev.o
CC util-linux/readprofile.o
CC util-linux/rtcwake.o
CC util-linux/script.o
CC util-linux/switch_root.o
CC util-linux/umount.o
AR util-linux/lib.a
LD util-linux/volume_id/built-in.o
CC util-linux/volume_id/get_devname.o
CC util-linux/volume_id/util.o
CC util-linux/volume_id/volume_id.o
AR util-linux/volume_id/lib.a
LINK busybox_unstripped
Trying libraries: crypt m
Library crypt is needed, can't exclude it (yet)
Library m is needed, can't exclude it (yet)
Final link with: crypt m
[root@tom busybox-1.13.3]# ls
applets Config.in fa.config loginutils networking sysklogd
arch console-tools findutils mailutils printutils testsuite
archival coreutils include Makefile procps TODO
AUTHORS debianutils init Makefile.custom README TODO_config_nommu
busybox docs INSTALL Makefile.flags runit util-linux
busybox_unstripped e2fsprogs libbb Makefile.help scripts
busybox_unstripped.map editors libpwdgrp miscutils selinux
busybox_unstripped.out examples LICENSE modutils shell
[root@tom busybox-1.13.3]#

8 Make File Image for Target File System

Please make sure you have installed “mktools” tools and have an image directory ready before continue

8.1 Make YAFFS2 Image

Enter “/opt/FriendlyARM/mini6410/linux” and execute the following command:

#mkyaffs2image-128M 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.

Note: you can make “rootfs_qtopia_qt4-s” a yaffs2 image too

Note: mkyaffs2image and mkyaffs2image-128M are only for SLC Nand Flash to make yaffs2 iamges and don’t apply to MLC2 Nand Flash.

8.2 Make UBIFS Image

Enter “/opt/FriendlyARM/mini6410/linux” and execute the following command:

#mkubimage-slc rootfs_qtopia_qt4 rootfs_qtopia_qt4-slc.ubi

This will compress the whole “rootfs_qtopia_qt4” into a UBIFS 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.

Note: you can make "rootfs_qtopia_qt4-s" a UBIFS image too

8.3 Make EXT3 Image

Enter "/opt/FriendlyARM/mini6410/linux" and execute the following command:

mkext3image rootfs_qtopia_qt4 rootfs_qtopia_qt4.ext3

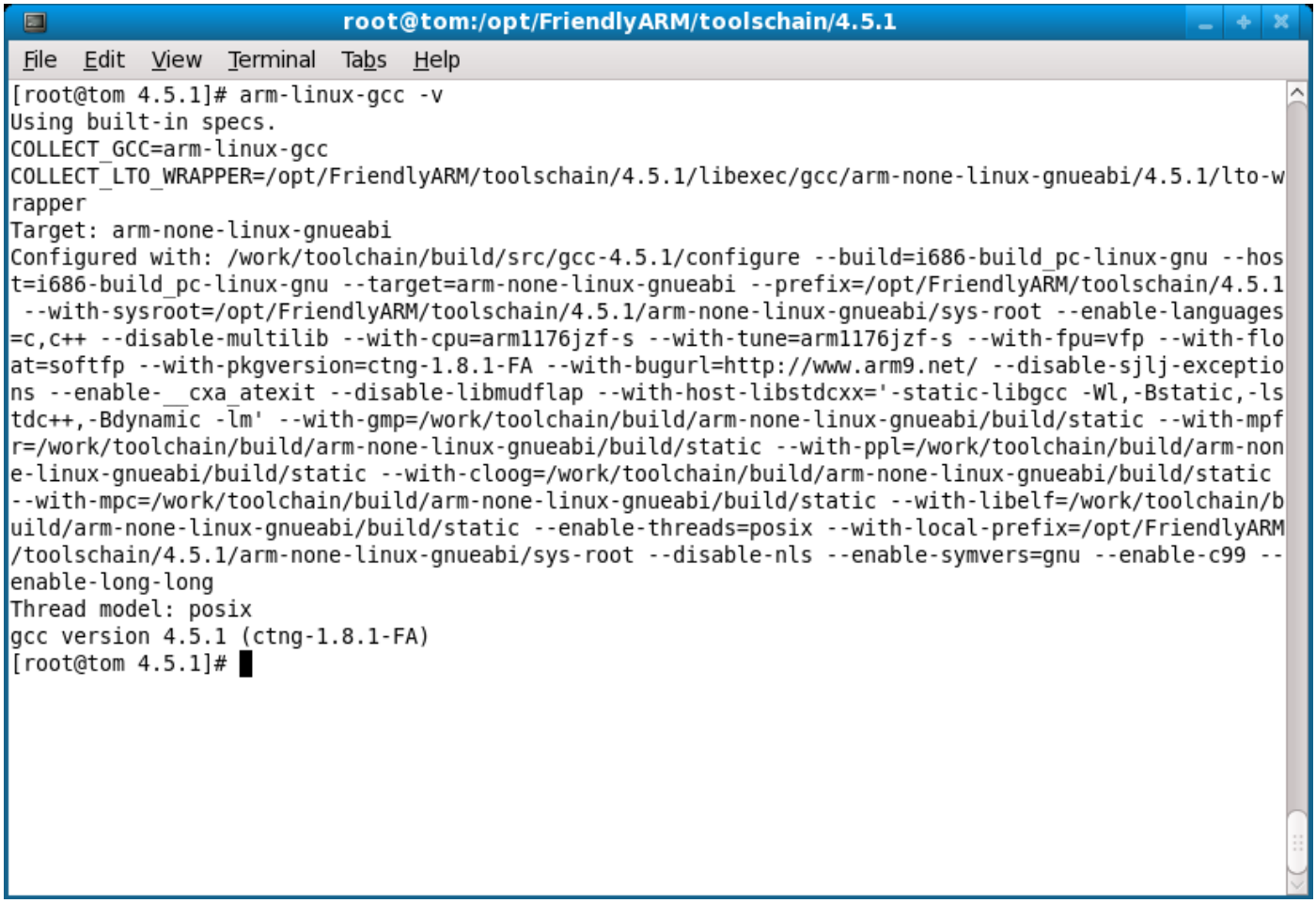
This will compress the whole "rootfs_qtopia_qt4" into a EXT3 rootfs_qtopia_qt4.img file. It is the same as the one in "/images/Linux/" in the shipped CD. Download it to your board's NAND Flash.

Note: you can make "rootfs_qtopia_qt4-s" a EXT3 image too

9 Sample Linux Programs

This section lists some sample Linux programs for users' reference.

You can find those programs under "**/opt/FriendlyARM/mini6410/examples**".All the following programs are compiled with arm-linux-gcc-4.5.1-v6-vfp. We don't guarantee they can be compiled and run with other corss compilers.



9.1 “Hello, World”

The source code of “Hello,World” is under “**/opt/FriendlyARM/mini6410/linux/examples/hello**”. Its contents are as follows:

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

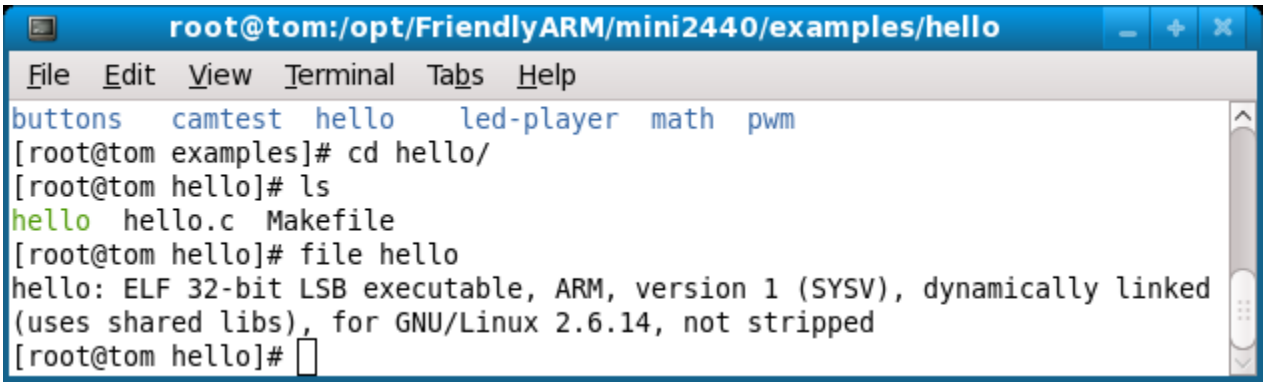
Step1: Compile Hello,World

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

#cd /opt/FriendlyARM/mini6410/linux/examples/hello

#make

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



Step2: Download “Hello,World” to Board

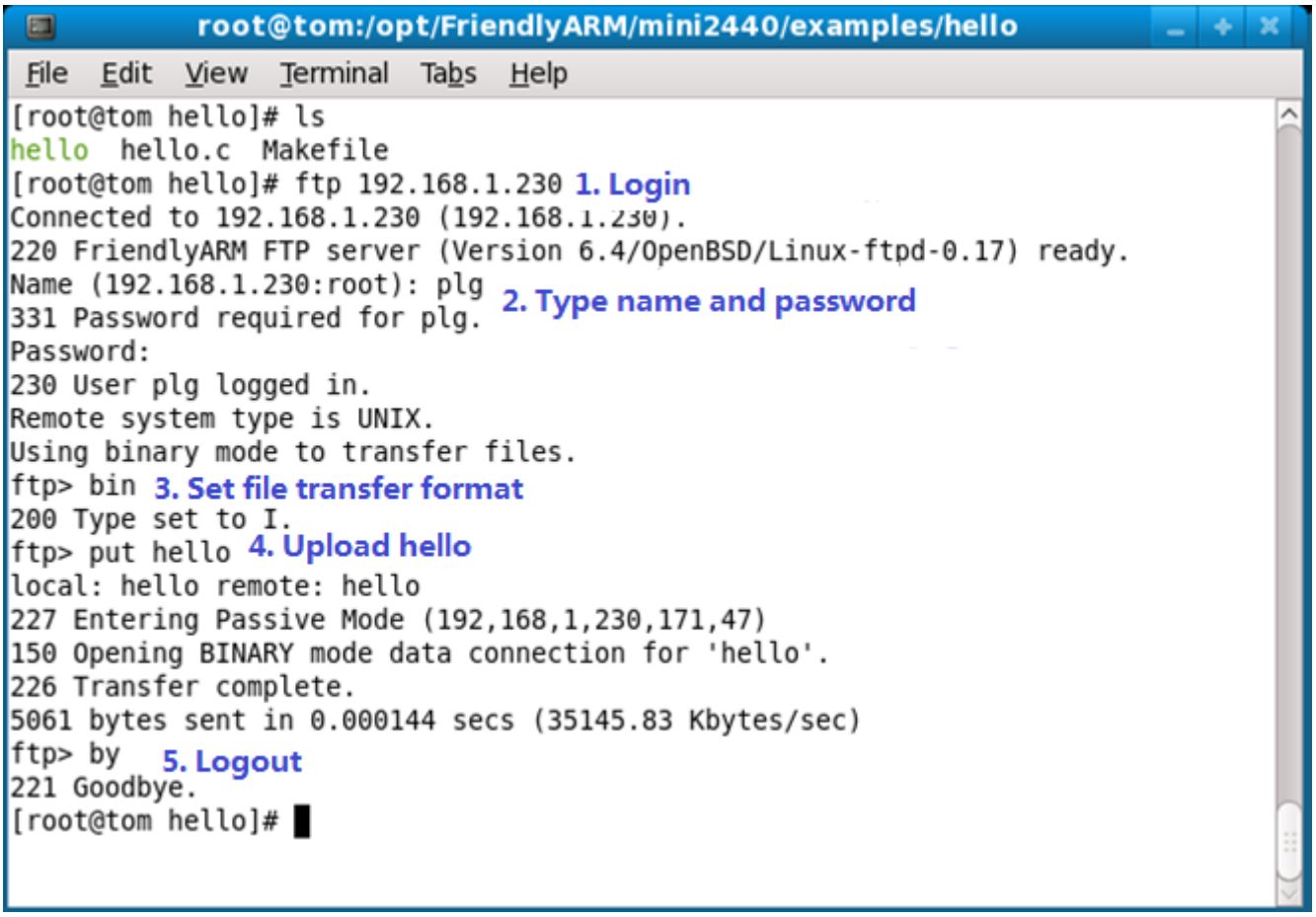
You can download your executable to the board in any of the following ways:

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

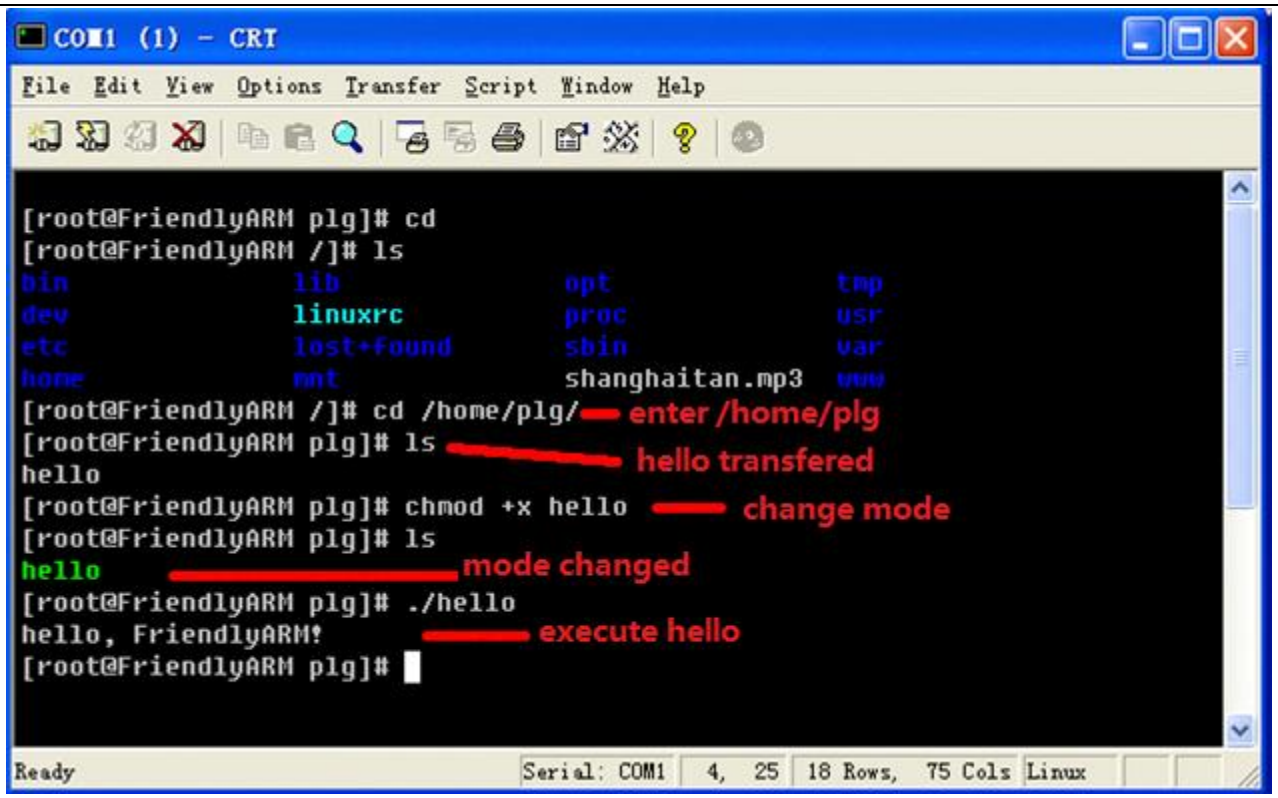
(1) FTP File Transfer

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:



(2) 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]# _
```

(3) File Transfer via Serial Port

Download your file to the board via serial port and change its property to executable

#chmod +x hello

Note: some users do this via a USB to Serial connector. This may not be successful due to the connector's

quality issues therefore we recommend file transfer via FTP

(4) Run via NFS

It is very popular to launch programs via NFS in Linux. This saves download time especially for large size files. Please set up your NFS server and type the commands below (in our example the IP was 192.168.1.111):

#mount –t nfs –o nolock

192.168.1.111:/opt/FriendlyARM/mini6410/linux/rootfs_qtopia_qt4 /mnt

If the mounting is successful you can enter "/mnt" to manipulate your files. Please copy "hello" to "**opt/FriendlyARM/mini6410/linux/rootfs_qtopia_qt4**" and type the following commands:

#cd /mnt

#./hello

9.2 LED Test Program

The source code of "Hello,World" is under "**/opt/FriendlyARM/mini6410/linux/examples/hello**". Its contents are as follows:

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/char
Driver	mini6410_leds.c
Device Type	misc

Device Name	/dev/leds
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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

9.3 User Button Test Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/char
Driver	mini6410_buttons.c
Device Type	misc
Device Name	/dev/buttons
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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

9.4 PWM Buzzer Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/char
Driver	mini6410_pwm.c
Device Type	misc
Device Name	/dev/pwm
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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

9.5 I2C-EEPROM Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/i2c/busses
Driver	I2c-s3c2410c
Device Type	Char
Device Name	/dev/i2c/0
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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

9.6 Serial Port Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/serial/
Driver	S3c6400.c
Device Type	
Device Name	/dev/ttySAC0, 1, 2 and 4
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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 FriendlyARM 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 FriendlyARM 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

9.7 UDP Program

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/net/
Driver	Dm9000.c
Device Type	
Device Name	eth0 (not listed in /dev)
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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 mini2440 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/mini6410/linux/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.

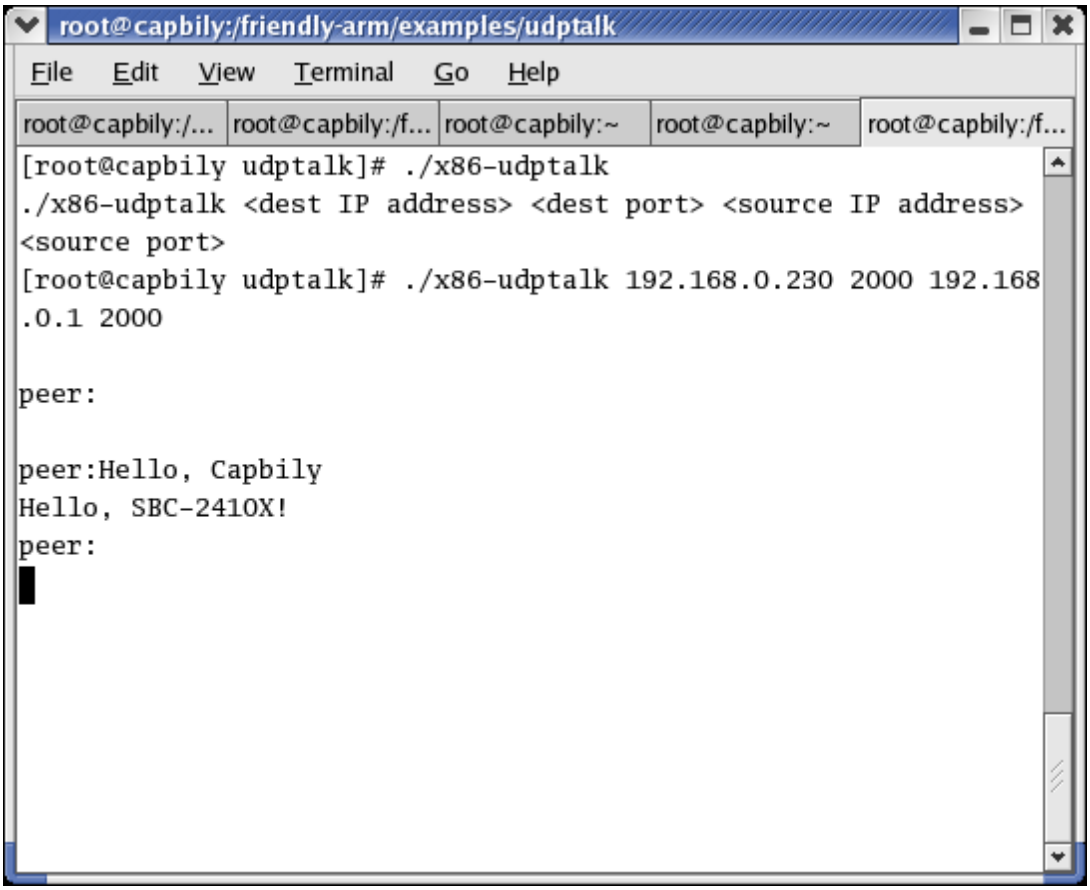
Type the following command on your host:

#./x86-udptalk 192.168.1.230 2000 192.168.1.108 2000

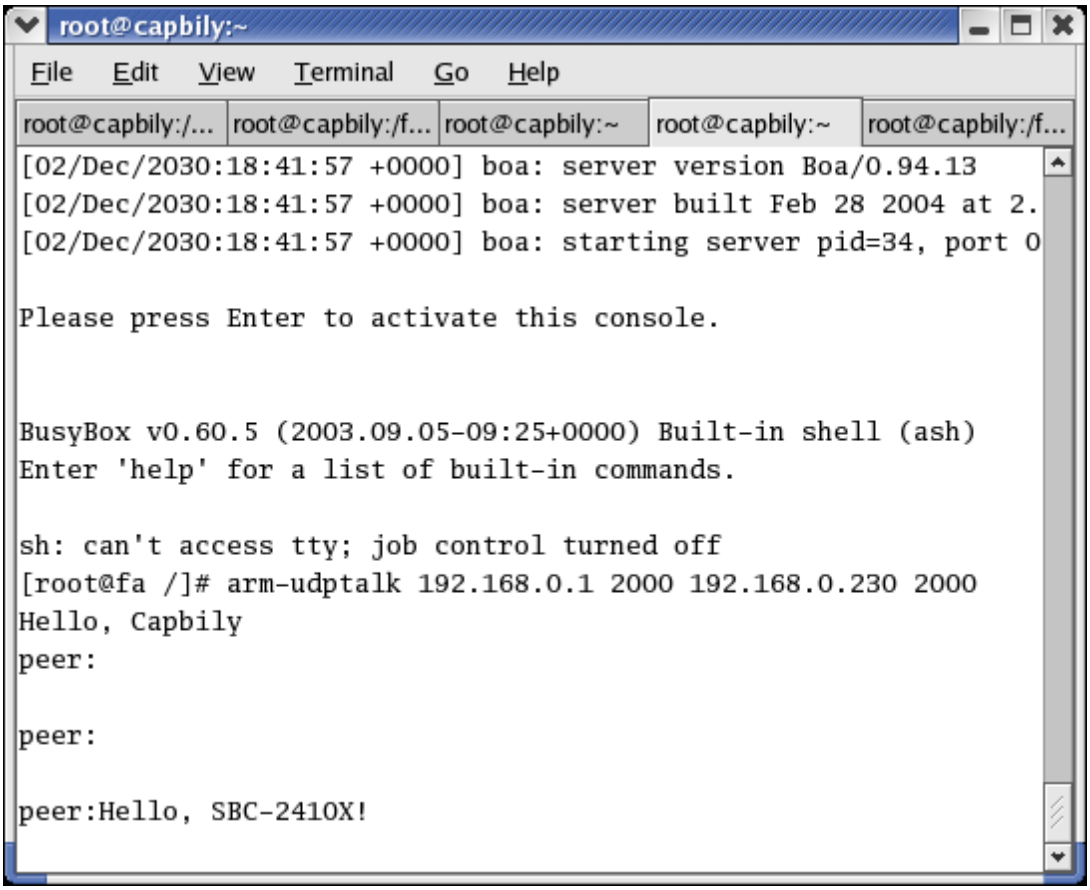
Type the following command on your board

#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

9.8 Utilize Math Libraries

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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 *~
```

9.9 Thread Programming

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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 *~
```

9.10 Pipe Programming – Manipulating LED from HTML

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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
```

9.11 C++ “Hello, World”

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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:";
}
```

10 Sample Linux Driver Programs

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

10.1 Hello Module

Program Description:	
Source Code Location	
Driver	
Device Type	
Device Name	
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/char
Test Program Name	Mini6410_hello_module.c
Executable Name	
Test Program’s Location in Board	
Note: mounting this driver will not create a device node under /dev	
Program:	

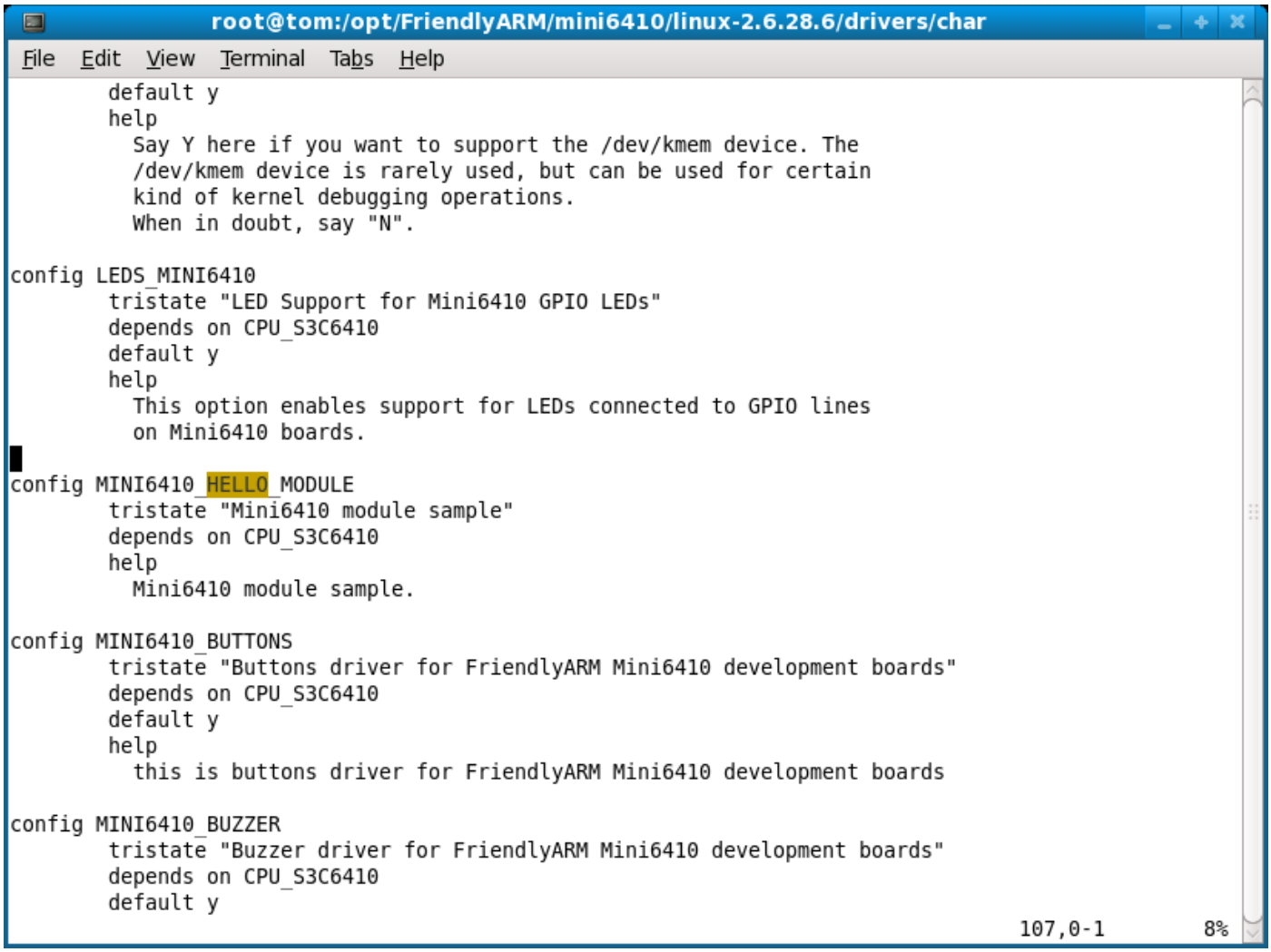
```
#include <linux/kernel.h>
#include <linux/module.h>
static int __init mini6410_hello_module_init(void)
{
printk("Hello, Mini6410 module is installed !\n");
return 0;
}
static void __exit mini6410_hello_module_cleanup(void)
{
printk("Good-bye, Mini6410 module was removed!\n");
}
module_init(mini6410_hello_module_init);
```

```
module_exit(mini6410_hello_module_cleanup);
MODULE_LICENSE("GPL");
```

(1) Integrate Hello,Module into Kernel and Compile

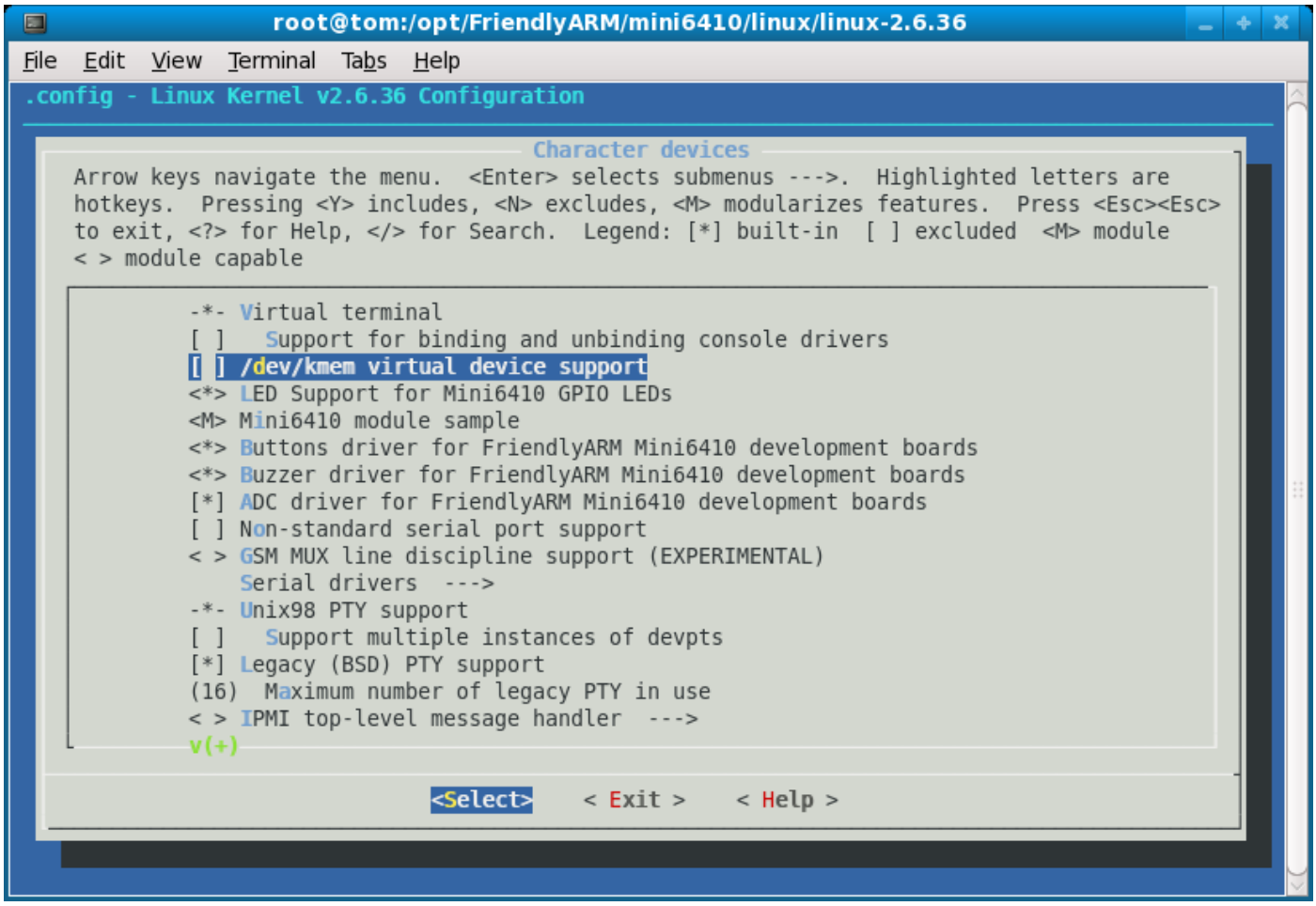
Please follow the steps below to include the module into the kernel and compile (Note: actually the following steps have been set up and you only need to directly compile it)：

Step1: configure “Kconfig”, add this module in the drivers and it will appear in make menuconfig. Open “linux-2.6.36/drivers/char/Kconfig” add lined marked in yellow

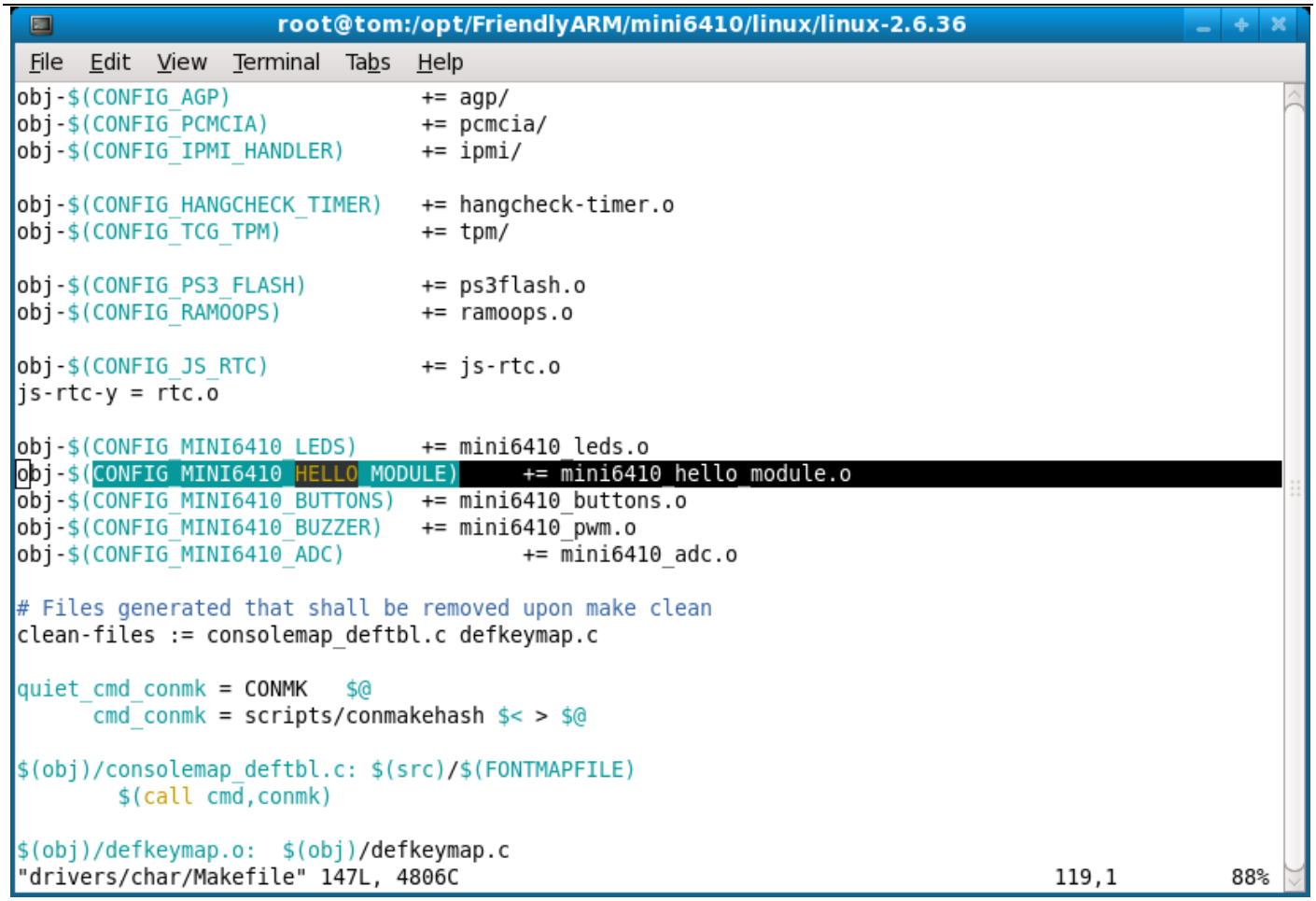


Save and exit. When you run “make menuconfig” in the linux-2.6.36 directory you will

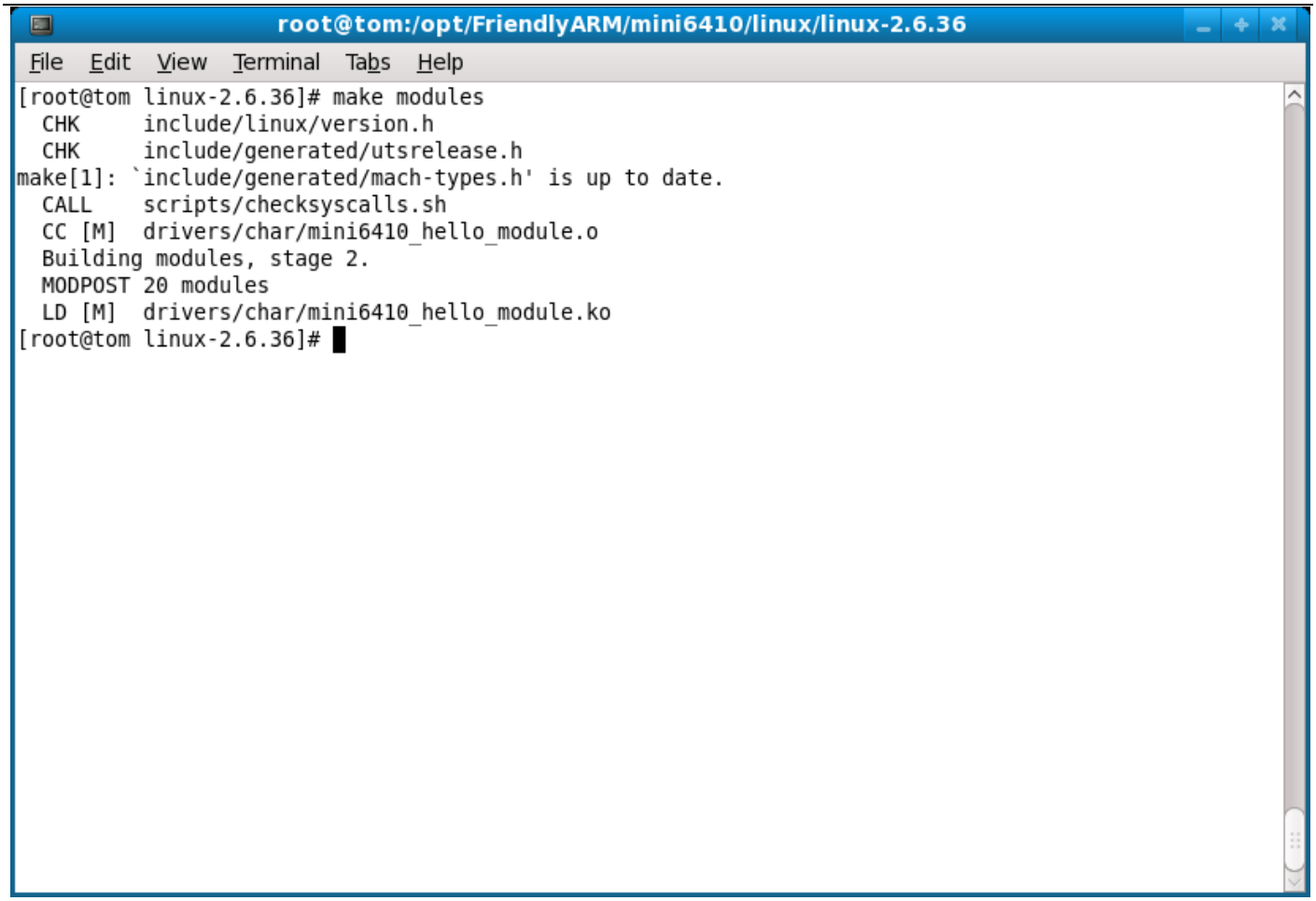
see your item in Device Drivers -> Character devices. Press the space key it will be marked "<M>". This means this souce code will be compiled as a module. Press the space key again it will be marked "<*>". This means it will be compiled into the kernel. Here we chose "<M>"



Step2: the previous step still cannot include it into the kernel when compiling. You need tolinkthe kernel configuration to the source code in "makefile". Open "linux-2.6.36/drivers/char/Makefile", add the marked line shown below, save and exit



Step3: go back to the linux-2.6.36 source code directory, run “make modules” a “mini6410_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.



(2) Download Hello, Module to Board

Please transfer "mini6410_hello_module.ko" to the board via FTP and move it to "**/lib/modules/2.6.36-FriendlyARM**"

#modprobe mini6410_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 and you will observe that the module has been unmounted

#rmmod mini6410_hello_module

Note: to load a module correctly, you need to move your module to the boards's "**/lib/modules/2.6.36-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/2.6.36-FriendlyARM.**

```
[root@FriendlyARM 2.6.36-FriendlyARM]# ls /home/plg/
mini6410_hello_module.ko
[root@FriendlyARM 2.6.36-FriendlyARM]# pwd
/lib/modules/2.6.36-FriendlyARM
[root@FriendlyARM 2.6.36-FriendlyARM]# cp /home/plg/mini6410_hello_module.ko .
[root@FriendlyARM 2.6.36-FriendlyARM]# ls
build                   modules.dep             modules.order
kernel                  modules.dep.bb          modules.pcimap
mini6410_hello_module.ko  modules.ieee1394map     modules.seriomap
modules.alias           modules.inputmap        modules.symbols
modules.builtin         modules.isapnpmap       modules.usbmap
modules.ccwmap          modules.ofmap           source
[root@FriendlyARM 2.6.36-FriendlyARM]# modprobe mini6410_hello_module
Hello, Mini6410 module is installed !
[root@FriendlyARM 2.6.36-FriendlyARM]# rmmod mini6410_hello_module
Good-bye, Mini6410 module was removed!
[root@FriendlyARM 2.6.36-FriendlyARM]#
```

10.2 LED Driver

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

LED	IO Register	CPU Pin
LED1	GPK4	R23

LED2	GPK5	R22
LED3	GPK6	R24
LED4	GPK7	R25

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/mini6410/linux/linux-xxx/drivers/char
Driver	Mini6410_leds.c
Device Type	Misc, auto generated
Device Name	/dev/leds
Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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:	

```
#include <linux/miscdevice.h>
#include <linux/delay.h>
#include <asm/irq.h>
//#include <mach/regs-gpio.h>
#include <mach/hardware.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/mm.h>
#include <linux/fs.h>
#include <linux/types.h>
#include <linux/delay.h>
#include <linux/moduleparam.h>
#include <linux/slab.h>
#include <linux/errno.h>
#include <linux/ioctl.h>
```

```
#include <linux/cdev.h>
#include <linux/string.h>
#include <linux/list.h>
#include <linux/pci.h>
#include <asm/uaccess.h>
#include <asm/atomic.h>
#include <asm/unistd.h>
#include <mach/map.h>
#include <mach/regs-clock.h>
#include <mach/regs-gpio.h>
#include <plat/gpio-cfg.h>

#include <mach/gpio-bank-e.h>
#include <mach/gpio-bank-k.h>
#define DEVICE_NAME "leds"
static long sbc2440_leds_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
switch(cmd) {
unsigned tmp;
case 0:
case 1:
if (arg > 4) {
return -EINVAL;
}
tmp = readl(S3C64XX_GPKDAT);
tmp &= ~(1 << (4 + arg));
tmp |= ( (!cmd) << (4 + arg) );
writel(tmp, S3C64XX_GPKDAT);
//printk (DEVICE_NAME": %d %d\n", arg, cmd);
return 0;
default:
return -EINVAL;
}
}
static struct file_operations dev_fops = {
.owner = THIS_MODULE,
.unlocked_ioctl = sbc2440_leds_ioctl,
};
static struct miscdevice misc = {
```

```
.minor = MISC_DYNAMIC_MINOR,
.name = DEVICE_NAME,
.fops = &dev_fops,
};
static int __init dev_init(void)
{
int ret;
{
unsigned tmp;
tmp = readl(S3C64XX_GPKCON);
tmp = (tmp & ~(0xffffU<<16))|(0x1111U<<16);
writel(tmp, S3C64XX_GPKCON);
tmp = readl(S3C64XX_GPKDAT);
tmp |= (0xF << 4);
writel(tmp, S3C64XX_GPKDAT);
}
ret = misc_register(&misc);
printk (DEVICE_NAME"\tinitialized\n");
return ret;
}
static void __exit dev_exit(void)
{
misc_deregister(&misc);
}
module_init(dev_init);
module_exit(dev_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("FriendlyARM Inc.");
```

10.3 Button Driver

Program Description:	
Source Code Location	/opt/FriendlyARM/mini6410/linux/linux-xxx/drivers/char
Driver	Mini6410_buttons.c
Device Type	Misc, auto generated
Device Name	/dev/buttons

Test Program Source Code Location	/opt/FriendlyARM/mini6410/linux/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	GPN0	EINT0
K2	GPN1	EINT1
K3	GPN2	EINT2
K4	GPN3	EINT3
K5	GPN4	EINT4
K6	GPN5	EINT5
K7	GPN6	EINT6
K8	GPN7	EINT7

Program:

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/poll.h>
#include <linux/irq.h>
#include <asm/irq.h>
#include <asm/io.h>
#include <linux/interrupt.h>
#include <asm/uaccess.h>
#include <mach/hardware.h>
#include <linux/platform_device.h>
#include <linux/cdev.h>
#include <linux/miscdevice.h>
#include <mach/map.h>
#include <mach/regs-clock.h>
#include <mach/regs-gpio.h>

#include <plat/gpio-cfg.h>
#include <mach/gpio-bank-n.h>
#include <mach/gpio-bank-l.h>
#define DEVICE_NAME "buttons"
struct button_irq_desc {
```

```
int irq;
int number;
char *name;
};
static struct button_irq_desc button_irqs [] = {
{IRQ_EINT( 0), 0, "KEY0"},
{IRQ_EINT( 1), 1, "KEY1"},
{IRQ_EINT( 2), 2, "KEY2"},
{IRQ_EINT( 3), 3, "KEY3"},
{IRQ_EINT( 4), 4, "KEY4"},
{IRQ_EINT( 5), 5, "KEY5"},
{IRQ_EINT(19), 6, "KEY6"},
{IRQ_EINT(20), 7, "KEY7"},
};
static volatile char key_values [] = {'0', '0', '0', '0', '0', '0', '0', '0'};
static DECLARE_WAIT_QUEUE_HEAD(button_waitq);
static volatile int ev_press = 0;
static irqreturn_t buttons_interrupt(int irq, void *dev_id)
{
struct button_irq_desc *button_irqs = (struct button_irq_desc *)dev_id;
int down;
int number;
unsigned tmp;
udelay(0);
number = button_irqs->number;
switch(number) {
case 0: case 1: case 2: case 3: case 4: case 5:
tmp = readl(S3C64XX_GPNDAT);
down = !(tmp & (1<<number));
break;
case 6: case 7:
tmp = readl(S3C64XX_GPLDAT);
down = !(tmp & (1 << (number + 5)));
break;
default:
down = 0;
}
if (down != (key_values[number] & 1)) {
key_values[number] = '0' + down;
```

```
ev_press = 1;
wake_up_interruptible(&button_waitq);
}
return IRQ_RETVAL(IRQ_HANDLED);
}
static int s3c64xx_buttons_open(struct inode *inode, struct file *file)
{
int i;
int err = 0;
for (i = 0; i < sizeof(button_irqs)/sizeof(button_irqs[0]); i++) {
if (button_irqs[i].irq < 0) {
continue;
}
err = request_irq(button_irqs[i].irq, buttons_interrupt, IRQ_TYPE_EDGE_BOTH,
button_irqs[i].name, (void *)&button_irqs[i]);
if (err)
break;
}
if (err) {
i--;
for (; i >= 0; i--) {
if (button_irqs[i].irq < 0) {
continue;
}
disable_irq(button_irqs[i].irq);
free_irq(button_irqs[i].irq, (void *)&button_irqs[i]);
}
return -EBUSY;

ev_press = 1;
return 0;
}
static int s3c64xx_buttons_close(struct inode *inode, struct file *file)
{
int i;
for (i = 0; i < sizeof(button_irqs)/sizeof(button_irqs[0]); i++) {
if (button_irqs[i].irq < 0) {
continue;
}
```

```
free_irq(button_irqs[i].irq, (void *)&button_irqs[i]);
}
return 0;
}
static int s3c64xx_buttons_read(struct file *filp, char __user *buff, size_t count, loff_t *offp)
{
unsigned long err;
if (!ev_press) {
if (filp->f_flags & O_NONBLOCK)
return -EAGAIN;
else
wait_event_interruptible(button_waitq, ev_press);
}
ev_press = 0;
err = copy_to_user((void *)buff, (const void *)(&key_values), min(sizeof(key_values), count));
return err ? -EFAULT : min(sizeof(key_values), count);
}
static unsigned int s3c64xx_buttons_poll( struct file *file, struct poll_table_struct *wait)
{
unsigned int mask = 0;
poll_wait(file, &button_waitq, wait);
if (ev_press)
mask |= POLLIN | POLLRDNORM;
return mask;
}
static struct file_operations dev_fops = {
.owner = THIS_MODULE,
.open = s3c64xx_buttons_open,
.release = s3c64xx_buttons_close,
.read = s3c64xx_buttons_read,
.poll = s3c64xx_buttons_poll,
};
static struct miscdevice misc = {
.minor = MISC_DYNAMIC_MINOR,
.name = DEVICE_NAME,
.fops = &dev_fops,
};
static int __init dev_init(void)
{
```

```
int ret;
ret = misc_register(&misc);
printk (DEVICE_NAME"\tinitialized\n");
return ret;
}
static void __exit dev_exit(void)
{
misc_deregister(&misc);
}
module_init(dev_init);
module_exit(dev_exit);
MODULE_LICENSE("GPL");
MODULE_AUTHOR("FriendlyARM Inc.");
```

11 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.

11.1 Uncompress and Install Source Code

Please refer to Section 4

11.2 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/mini6410/linux/x86-qtopia

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

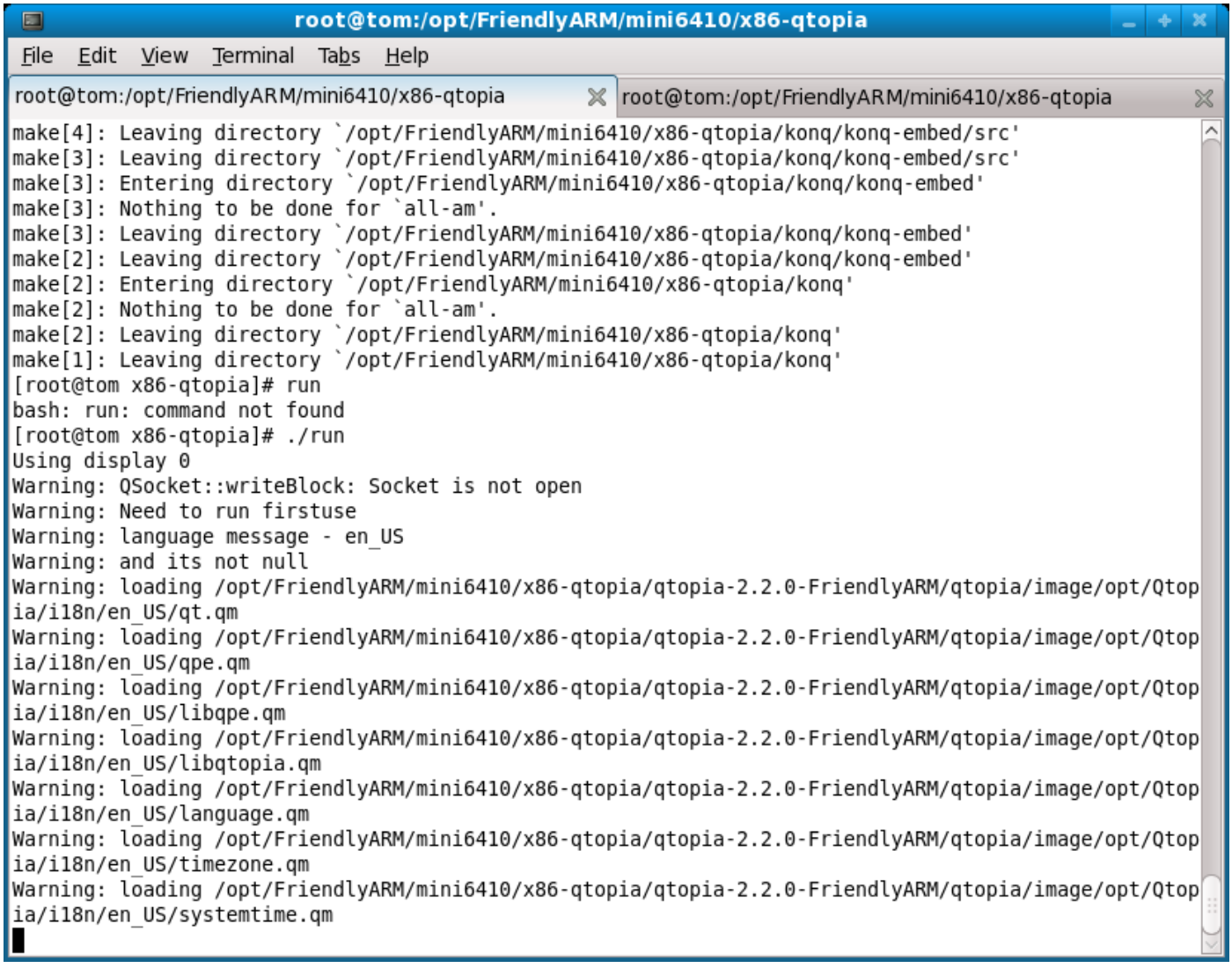
Note: **./build-all** will automatically compile the complete Qtopia and its embedded web browser. You can execute "**./build**" first and then "./**build-konq**" to compile them separately.

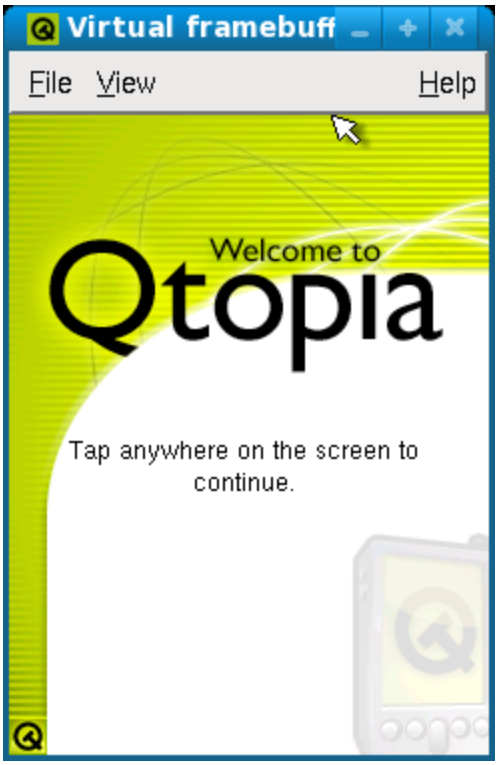
To run your qtopia you can type the

command below:

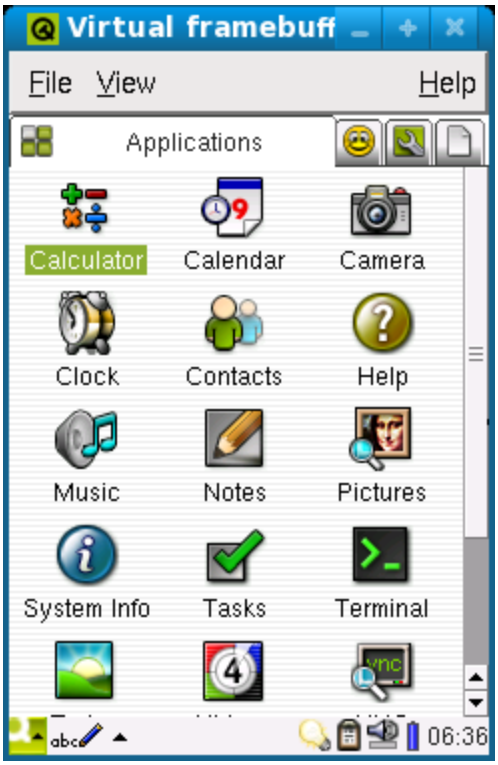
#./**run**

You will see the following screen





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



11.3 Compile and Run Qtopia-2.2.0 for ARM

Please make sure your compiler is arm-linux-gcc-4.4.1 and platform is Fedora 9. Enter the working directory and type the command below

#cd /opt/FriendlyARM/mini6410/linux/arm-qtopia

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

#./mktarget (this makes a file system image and will generate "**target-qtopia-konq.tgz**")

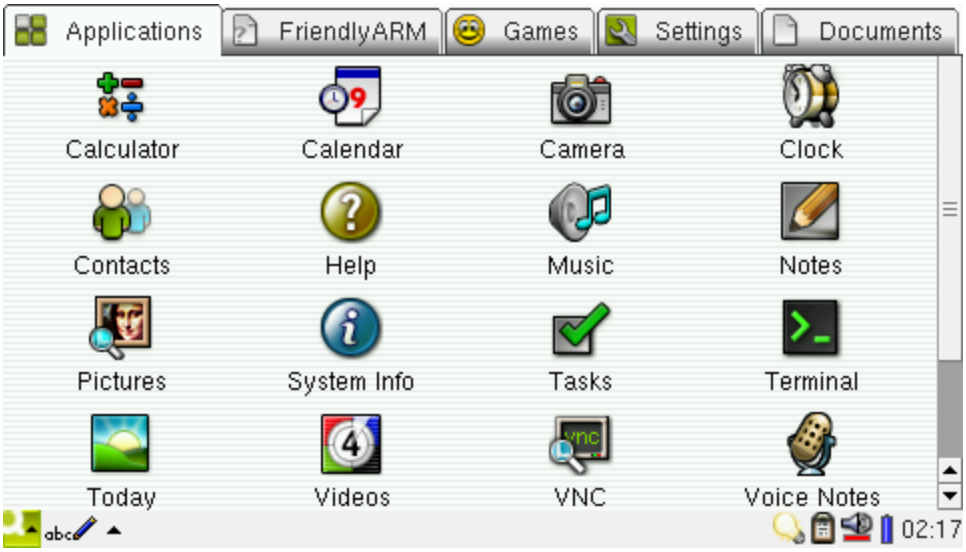
Note: "**./build-all**" will automatically compile a complete Qtopia system and the web browser and generate Jpeg, GIF, PNG image files. You can execute "**./build**" first and then "./**build-konq**" to compile them separately.

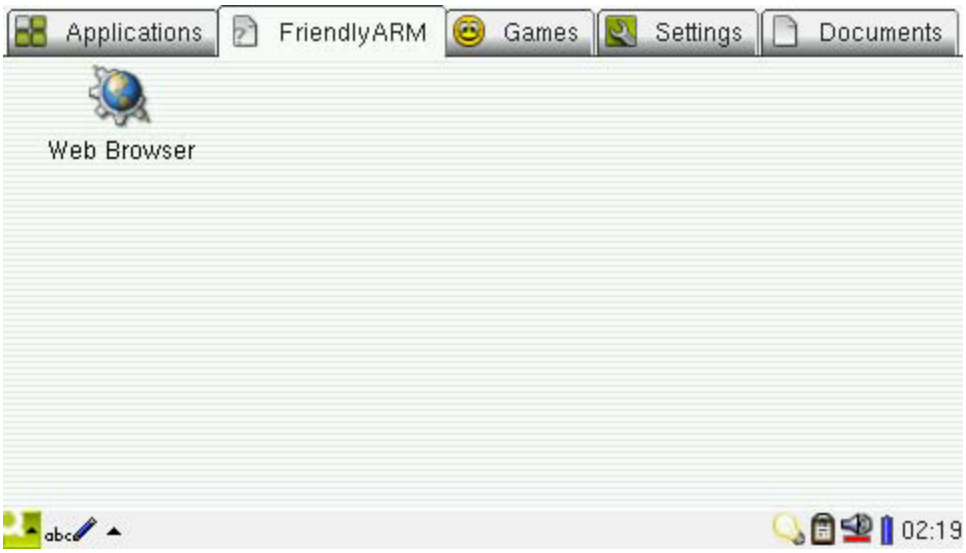
To remove your old Qtopia system you just need to delete all the files under "/opt". Then you can uncompress your target-qtopia-konq.tgz to the board's root directory via a flash drive. In our example we had it under /home/plg. Please run the command below:

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

"C" means "Change" and "/" after "C" means it will be uncompressed to the root directory. Afteryouare 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

12 Compile QtE-4.7.0

12.1 Uncompress and Install Source Code

Please refer to section 4.1

12.2 Compile and Run QtE-4.7.0 for ARM

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build-all script for users to easily compile QtE-4.7.0. Please enter the source code directory and type the following command:

#cd /opt/FriendlyARM/mini6410/linux/arm-qte-4.7.0

#./build-all

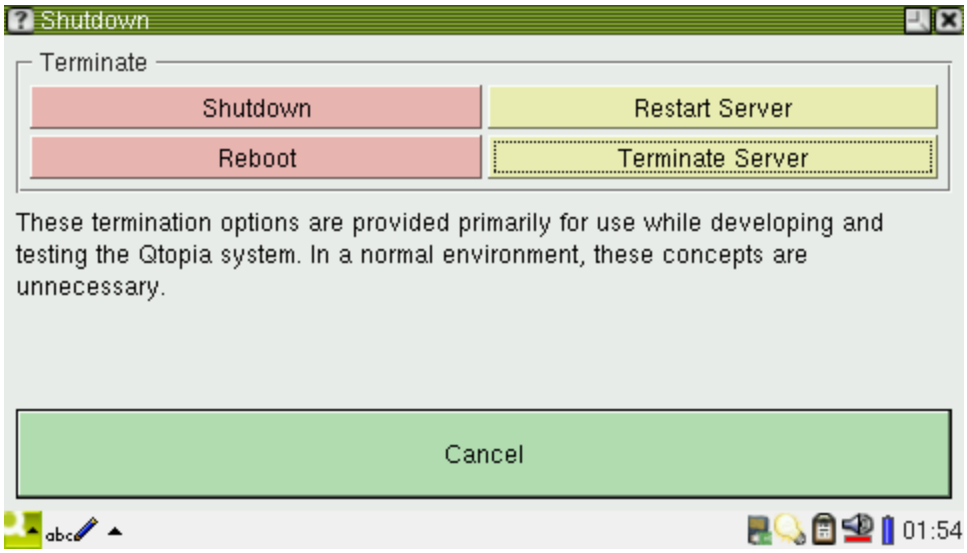
The build process takes a while. And after it is done, please run the mktarget script and a **target-qte-4.7.0.tgz** will be generated. Please follow the command below:

#tar xvzf target-qte-4.7.0.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-4.7.0, to test your build you can delete the one on your board by "rm" the whole "**/usr/local/Trolltech**" directory.

Before running QtE-4..6.3, 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"

13 Compile Qtopia4(Qt-Extended-4.4.3)

13.1 Uncompress and Install Source Code

Please refer to section 4.1

13.2 Compile and Run Qt-Extened-4.4.3 for x86

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build-all script for users to easily compile Qt-Extened-4.4.3. Please enter the source code directory and type the following command:

#cd /opt/FriendlyARM/mini6410/linux/x86-qt-extended-4.4.3

#./build-all

The build process takes a while. To run your compiled system please type the command below:

#./run

Now you will see the following screen:


Virtual framebuffer 240x320 16bpp Display :0 Rotate (
Welcome to
Qt Extended
Press Select to continue.
Qt
1 oo
2 abc
3 def
4 ghi
5 jkl
6 mno
7 pqrs
8 tuv
9 wxyz
*
0
#


Virtual framebuffer 240x320 16bpp Display :0 Rotate (
6:01
Contacts
Options
Select
Back
Qt
1 oo
2 abc
3 def
4 ghi
5 jkl
6 mno
7 pqrs
8 tuv
9 wxyz
*
0
#

13.3 Compile and Run Qt-Extened-4.4.3 for ARM

Note: please use our arm-linux-gcc-4.5.1 and Fedora9 to compile. We offered a build-all script for users to easily compile Qt-Extened-4.4.3. Please enter the source code directory and type the following command:

#cd /opt/FriendlyARM/mini6410/linux/arm-qt-extended-4.4.3

#./build-all

The build process takes a while. To run your compiled system please type the command below:

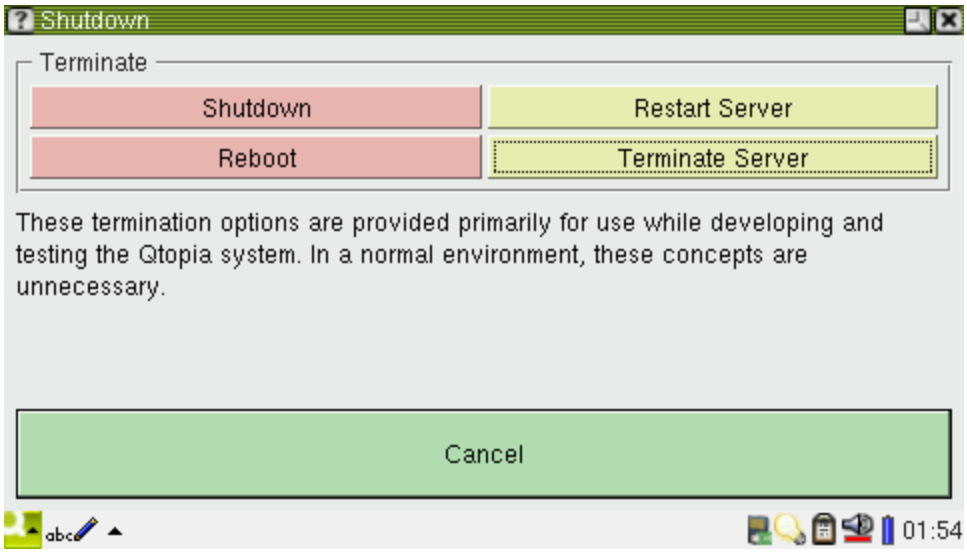
#./**run**

And after it is done, please run the mktarget script and a **target-qtopia4.tgz** will be generated. Please follow the command below:

#tar xvzf target-qtopia4.tgz –C /

A Qtopia4.4.3 directory will be generated under "/opt", which includes all needed libraries and executables. Since our shipped Linux already includes QtE-4.7.0, to test your build you can delete the one on your board by "rm" the whole "**/opt/Qtopia4.4.3**" directory.

Before running Qtopia4, 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 "qtopia4 &"